Discrete Modelling of Vehicular Traffic and Pedestrian Dynamics from a Technical Perspective

vorgelegt von

Dipl.-Inf.
Minjie Chen
geb. in Shanghai, China

von der Fakultät II - Mathematik und Naturwissenschaften
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
- Dr.-Ing. -

genehmigte Dissertation

Promotionsausschuss:

Vorsitzender: Prof. Dr. Michael Scheutzow
Gutachter: Prof. Dr. Günter Bärwolff
Gutachter: Prof. Dr. Reinhard Nabben
Gutachter: Prof. Dr. Armin Seyfried
(Jülich Supercomputing Centre)

Tag der wissenschaftlichen Aussprache: 05.07.2017

Berlin 2017

Zusammenfassung

Die vorliegende Arbeit präsentiert einige Modellierungsansätze für Verkehrsdynamik und Personendynamik aus der mikroskopischen Kategorie. Verkehrsdynamik und Personendynamik sind zwei in den letzten Jahrzehnten neu entstandene interdisziplinäre Forschungsgebiete, die zwar unterschiedliche Themen behandeln, aber auch gewisse Gemeinsamkeiten aufweisen: Erstens unterliegen beide bestimmten von Menschen definierten Verhaltensregeln; zweitens können sowohl Verkehrsdynamik als auch Personendynamik aus technischer Sicht auf sehr ähnlichen geometrischen Strukturen mikroskopisch modelliert werden. Derartige geometrische Strukturen werden für die hier vorgestellten Methoden verwendet. Diese Arbeit ist in vier Teile gegliedert. In Kapitel 1 wird nach einer kurzen Einführung in makroskopische Modellierung einige wichtige Begriffe erkläutert, die in allen gängigen Methoden eine zentrale Bedeutung haben. In Kapitel 2 werden zuerst zwei bekannte Zellulärer-Automat-Modelle für Verkehrsdynamik beschrieben. Als Erweiterung auf diskreten Geometrien wird ein neuer Ansatz für mehrspurigen Straßenverkehr vorgestellt, begleitet von ausführlichen Kalibrierungsdetails. Kapitel 3 behandelt die Thematik der Personendynamik. Hier wird der nicht mehr reduzierbare Raumanspruch der einzelnen Personen als Teilnehmer in einem interaktiven System besonders beachtet, während jene in makroskopischen Ansätzen üblicherweise als Partikeln gesehen werden (also nur über vernachlässigbare Volumina verfügen, was der Realität widerspricht). Mit Hilfe eines neuen Balancierungsmechanismus und einer lokalen Navigation, die nach einem kontinuierlichen Dichteschätzer aufgebaut ist, wird es ermöglicht, realitätsnahe Simulationsergebnisse zu produzieren. Mikroskopische Modellierung von Personendynamik ist zum Teil auch auf Modelle der zellulären Automaten zurückzuführen. In diesem Kapitel werden sowohl reguläre Geometrien, darstellbar durch die üblichen Rechteckgitter, als auch irreguläre Geometrien behandelt. Die Letzteren können auf den sogenannten pfad-orientierten Koordinatensystemen approximiert werden. Dabei wird eine Lösung für die Simulation auf nicht regulären Geometrien angeboten, welche sich wiederum in ein größeres System mit einer regulären Struktur einbetten lässt. In Kapitel 4 stellen wir zwei Verfahren zur automatischen Generierung von Trajektorien vor: Das erste ist eine photogrammetrische Methode für die Gewinnung von räumlichen Informationen aus konventionellen Videodaten mit Hilfe von vordefinierten Referenzpunkten. Diese Methode ist als Lösung eines Optimierungsproblems mit den von den Referenzen angegebenen Hilfsvariablen zu verstehen. Die in § 3.1.6 verwendeten empirischen Daten wurden in der Tat mittels dieser Methode gewonnen. Das zweite Verfahren kann für die dreidimensionalen, sogenannten Time-Of-Flight (TOF)-Daten verwendet werden. Unsere auf TOF basierende Methode kann auch für die automatische Fahrgastzählung eingesetzt werden; das Ergebnis ist der Leistung eines zu Vergleichszwecken herangezogenen marktüblichen industriellen Messgeräts leicht überlegen. Mit der automatischen Verarbeitung der Video- bzw. TOF-Daten wird selbstverständlich erheblicher menschlicher Arbeitseinsatz eingespart. Die dabei resultierenden Trajektorien sind für die übliche Kalibrierungsarbeit der unterschiedlichen Modellierungsmethoden sehr nützlich oder gar unerlässlich. Im Nachtrag werden alle hier entwickelten Rechenprogramme vorgestellt und ihre Nutzung wird kurz erläutert.

Abstract

Transportation systems are to be designed and implemented for a safe and efficient usage. In the past decades, vehicular traffic systems and pedestrian dynamics have emerged as two new interdisciplinary research fields. Although seemingly different, in both cases, human factors play a significant role; in addition, both vehicular traffic and pedestrian dynamics can be modelled from a same discrete perspective and they can be formulated on similar background geometries. The current dissertation presents a couple of discrete modelling methods in this aspect. Chapter 1 gives a brief discussion in general about macroscopic modelling in the two fields. Important terms in the two fields are explained as well. In Chapter 2, two cellular automaton models for the simulation of vehicular traffic are mentioned, on the same discrete geometry, we present a new deductive model for the simulation of vehicular traffic with multiple lanes and additional safety distance control, this model is accompanied by detailed calibration information. Chapter 3 deals with the topic of pedestrian dynamics. In macroscopic methods, pedestrians are mostly treated as particles with neglectable sizes, in our microscopic modelling, however, an exclusive personal space for the pedestrians is considered. In correspondence to this, a new balancing mechanism and a local navigation method based on continuous density are proposed to describe the interaction among the pedestrians. Further considerations have been given to the underlying geometries. Common cellular automaton models are defined on square grids, in comparison, we offer an approximation scheme of the so-called path-oriented coordinate systems. By this means, irregular geometric settings which we sometimes encounter in real-world situations can be formulated. At the same time, such systems can be later embedded into larger and more complex simulation systems defined on regular grids. In Chapter 4 we consider the problem of pedestrian trajectory generation and give two possible solutions. The first one is a photogrammetric method endeavoured to extract spacial information from conventional video data with the help of additional reference points in the scene, this method is essentially an optimization problem respecting the given reference points. In fact, the empirical data used for the model calibration in § 3.1.6 have been obtained by using this method. The second method refers to the so-called three-dimensional Time-Of-Flight data. Automatic detection of pedestrians as moving objects is realized and this method can be deployed for the purpose of automatic passenger counting in public transportation. With automatic processing of pedestrian trajectories, enormous amount of manual effort can be saved; on the other hand, trajectory information concerning real pedestrians in various scenarios can be very useful or even indispensable for the calibration of any existing simulation model in this field. In Appendix, the structure and usage of the related computer programs will be explained.

Acknowledgements

First of all, I wish to express my sincere gratitude to Prof. Günter Bärwolff for the guidance and supervision of this work. To Prof. Hartmut Schwandt I am deeply grateful for his support of my work and the constructive comments which improved the quality of this work substantially. I would also like to thank Prof. Reinhard Nabben and Prof. Armin Seyfried for reviewing my thesis and Prof. Michael Scheutzow for presiding over the reviewing process.

In addition, I would like to thank Prof. Bärwolff and Prof. Schwandt for their leadership of our research team over the years. I thank Dr. Matthias Plaue (former working colleague), Olga and Tatjana Ebers, Denis Klauser and Jan Sablatnig for their cooperation in our various research projects and those interesting discussions.

To my wife Marina I am indebted for her enormous patience while I was preparing this text or re-inventing my own wheels.

Finally, I would like to mention the name of my first real teacher, Mr. Jianzhong Jin. Mr. Jin, who later became the headmaster of Shanghai Cao Guangbiao Primary School, taught me and many others with the heart of a true mentor. To him I owe a lot.

Minjie Chen

Contents

List of Figures

Preface

The last one hundred and fifty years have witnessed the growth of human society on an explosive scale. This can be seen in rapid human population growth and the volume of scientific knowledge acquired by humans. The growth of the human population, which can be locally approximated by an exponential function, is at the same time an important motivation for scientific and technological advancement, it requires the development of social structure at a proper and corresponding pace to maintain the operational functions in human society. Social infrastructure, in the form of open facilities to the public, has much to deal with human factors in its planning, design and maintenance.

As an indispensable element in the human society, public transport evolves constantly in quality and capacity to meet the need of humans. A major part of public and private transportation is undertaken by wheeled and railed vehicles. Indoors, people are now sheltered in architectural structures much better constructed and more complicated than those a century ago. During the last three decades, vehicular traffic (in the context of wheeled vehicles) and pedestrian dynamics have became two new interdisciplinary research fields (see [23, 61, 71, 72, 14]), which is itself evidence of the apparently and presumably exponential growth of the total knowledge inventory of humans. These two, however differing from each other, share an intrinsic similarity: in the former, automobiles (and other mobile machines) under human control[1] build up the main components of the system, whereas in the latter, the role of the participants in the system is played directly by humans. In both cases, control rules defined by humans are implied.

The objective of the study of vehicular traffic is to make optimal utilization of the existent resources such as roads and highways with fixed capacities via effective planning and valid prediction, and at the same time, to ensure safety on a reasonable or pre-defined level. Owing to the swift development of the so-called "smart mobility"[2], research of vehicular traffic dynamics is flourishing. Research of pedestrian dynamics serves a similar purpose with human beings becoming the direct objects under study; a specific topic in the study of pedestrian dynamics is the egress (evacuation) problem. Nowadays evacuation is by governmental regulation to be given due consideration in the design and planning of eventually all facilities.

[1] We disregard the so-called "autonomic driving", even if this technique involves no human effort in its operation, it is still under simulated human control.

[2] This term refers to the concept of intelligent networking and control of the participating—often in an ad hoc manner—vehicles in a traffic system to achieve better fuel efficiency, optimal utilization of road resources, higher safety and comfortability (for human participants) etc.

A second similarity between the two is that vehicular and pedestrian dynamics can be modelled from the same discrete perspective and they can be formulated on similar background geometries. This is also the starting point of the current work.

This text is in fact the result of my work on discrete modelling of vehicular traffic and pedestrian dynamics at Technische Universität Berlin. Chapter 1 gives a brief introduction to both macroscopic and microscopic modelling of vehicular traffic and pedestrian dynamics. Important terminology is explained as well.

Chapter 2 starts with the theory of cellular automata. Thanks to their simplicity, cellular automaton models had been applied in the simulation of vehicular traffic. The underlying geometry of cellular automaton models can be defined on regular grids. In the current text, geometry on which a simulation system is to be constructed plays a significant role. Here we propose a new deductive model for vehicular traffic of multiple lanes on rectangular grids. The model describes both asymmetric and symmetric traffic systems. Unlike other established models in this category, our model takes into consideration additional safety distance control and the physical aspect of the vehicles, the latter of which is often neglected in microscopic modelling. Empirical parameters confirmed by the common literature have been adopted for the calibration of our model.

Chapter 3 is about discrete modelling of pedestrian dynamics. Numerous microscopic models—including cellular automaton models—had been built on square grids (or other similar regular grids). It is well-known that each pedestrian requires an individual exclusive personal space and in this sense, pedestrians can only be treated like particles within a certain limit of abstraction (the same goes for the modelling of vehicles). This personal space becomes the guideline for the component size of the square grid of the underlying geometry in our simulation. We also present a new balancing mechanism to describe the interaction among pedestrians with heterogeneous characteristics. Another contribution is the continuous density. Applied in our discrete space of modelling, continuous density has been used to define overlay functions, which in turn have been deployed for navigation of the pedestrians on the tactical level. Real-world video recordings have been used for model calibration on the square grid. The second part of this chapter proposes a new approximation scheme on the so-called path-oriented coordinate systems. Sometimes a square or regular grid proves to be awkward for the simulation of pedestrian dynamics, this is especially the case when the primary walking direction of the pedestrian is not a straight line. Our solution is to divide a given path into segments which will be locally approximated by arc sectors. The modelling on the local arc sectors will have an ordered triangular grid as background geometry. Pedestrians' position transitions on these local segments will be studied. The concept of continuous density evaluated in discrete space and the balancing mechanism are applicable, too. Simulation tests

of closed pedestrian flow on a ring-shaped area have been conducted. Generally speaking, a path-oriented coordinate system can be embedded as a local component into larger and more complex simulation systems.

Chapter 4 deals with information extraction from video recordings. We present a photogrammetric method to extract spacial information from conventional video recordings. This special problem of "incrementing" data of two dimensions to three dimensions is solved as an optimization problem with the knowledge of additional reference points in the scene. The empirical data used for the model calibration on the square grid in § 3.1.6 have been retrieved from raw video recordings using this method. Our second approach is related to the Time-Of-Flight (TOF) technology. We propose a collection of new data structures for the processing of TOF data. Interestingly, these data structures are defined on two-dimensional input data, which in turn bear a resemblance to the geometry on square grids. With the help of a simple Kalman filter, the new data structures will be applied to generate trajectories from TOF data recordings. Unfortunately, part of the available test data is of rather poor quality, data evaluation would even be a strenuous task for humans, nevertheless our procedure achieves better results than a well-established industrial instrument currently in use. Both methods in this chapter can be used for automatic detection and positioning of pedestrians as moving objects. Hence quantified estimates of real-world scenarios can be obtained. A successful and fully automated solution of this task can save a huge amount of human effort in video data evaluation. In fact the latter is exactly what is the most valuable for the calibration of the simulation models in this field, including those described in the previous chapter.

All computer programs contained in this text have been rewritten in the years 2015–2016. Over the years all the results have undergone various improvements, both conceptually and technically, in addition to the various less obvious but yet crucial minor corrections. The implementation has been therefore aimed to render a precise demonstration of the work described here. Explanation about structure and usage of the computer programs is given in the appendix. All illustrations in the current text (apart from the video frames in Figures 3.12 and 3.13) have been carefully generated as high-quality vector graphics, if any of their details are not fully identifiable in the printed version of this text, readers are encouraged to consult the electronic version of this text.

Chapter 1

Introductory notes on modelling

Vehicular traffic and pedestrian dynamics can be modelled in a macroscopic or microscopic manner. In both cases, physical movement of the objects in the system plays a key role in the modelling. Here we will first have a brief look of the macroscopic methods.

1.1 Macroscopic modelling of vehicular traffic

Macroscopic methods study the system dynamics from a collective viewpoint without distinguishing the actual constituent objects in the system.

Let X and t denote spatial position and time respectively. In fluid dynamics, the law of *mass conservation* has the following differential form:

$$\frac{\partial \varrho(X,t)}{\partial t} + \nabla \cdot q(X,t) = 0, \tag{1.1}$$

with density $\varrho(X,t)$, velocity $v(X,t)$ and $q(X,t)$ defined as the product of the former two:

$$q(X,t) = \varrho(X,t)v(X,t),$$

which is called *flow* (or *flux*). In both cases of vehicular traffic and pedestrian dynamics, the notion of flow is of a particular importance, since by flow volume the structural capacity of the system being modelled can be measured and quantified.

Macroscopic modelling of traffic flow is based on the similarity between the flow of vehicles and that of fluid. Physical quantities derivable from empirical data such as density and speed (or velocity) are assumed to be differentiable functions of space and time. If lane changing is not considered, the geometry of vehicular traffic flow has a one-dimensional nature. Writing x for X in such a case, (1.1) becomes

$$\frac{\partial \varrho(x,t)}{\partial t} + \frac{\partial q(x,t)}{\partial x} = 0 \tag{1.2}$$

with

$$q(x,t) = \varrho(x,t)v(x,t). \tag{1.3}$$

(1.2) can be seen as the mass conservation of vehicles in a closed traffic lane segment.

1.1.1 First-order case

Macroscopic modelling of vehicular traffic can be traced back to the Lighthill-Whitham-Richards model [41, 57]. This model assumes that average speed $v(x,t)$ is a function of traffic density $\varrho(x,t)$:

$$v(x,t) = v^{(0)}(\varrho(x,t)), \tag{1.4}$$

where $v^{(0)}$ stands for a function which delivers the expected speed value respecting a given density.[1] The function $v^{(0)}$ can be approximated from empirical data. The Lighthill-Whitham-Richards model is established by the first-order partial differential equation (1.2) combined with (1.3) and (1.4). The relationship (1.4) is called *fundamental diagram* of density and speed (cf. discussion in § 2.3.5.3). [70] contains a detailed comparison among the various types of fundamental diagram.

1.1.2 Second-order case and beyond

The Payne model [32] assumes the following relationship in the flow direction:

$$v(x(t+\Delta t), t+\Delta t) = v^{(0)}(\varrho(x+d), t), \tag{1.5}$$

where Δt refers to the reaction time of (the driver of) the vehicle currently at position $x(t)$ and d is the distance to the immediate leading vehicle in the flow direction. We notice that $\frac{1}{d}$ serves as an approximation of the local density:

$$\varrho(x,t) \doteq \frac{1}{d}. \tag{1.6}$$

(1.5) can be seen as a correction of the desired (or unimpeded) speed imposed by the current traffic conditions. The left side of (1.5) can be approximated by first-order Taylor expansion:

$$\begin{aligned} v(x(t+\Delta t), t+\Delta t) &\doteq v(x,t) + (x(t+\Delta t) - x(t))\frac{\partial v(x,t)}{\partial x} + \Delta t \cdot \frac{\partial v(x,t)}{\partial t} \\ &\doteq v(x,t) + \Delta t \cdot v(x,t)\frac{\partial v(x,t)}{\partial x} + \Delta t \cdot \frac{\partial v(x,t)}{\partial t}. \end{aligned} \tag{1.7}$$

Similarly, the right side of (1.5) can be approximated by:

$$v^{(0)}(\varrho(x+d),t) \doteq v^{(0)}(\varrho(x,t)) + d \cdot \frac{\partial \varrho(x,t)}{\partial x} \cdot \frac{\mathrm{d}v^{(0)}(\varrho(x,t))}{\mathrm{d}\varrho}$$

[1] The term "expected speed value" is sometimes called "equilibrium velocity" or "unimpeded speed", and $v^{(0)}$ is often written as v^e or V^e. Here we adopt the superscript (0) to preserve a uniform notation with that later in § 1.3.2.

and with (1.6)

$$\doteq v^{(0)}(\varrho(x,t))+\frac{1}{\varrho(x,t)}\cdot\frac{\partial\varrho(x,t)}{\partial x}\cdot\frac{\mathrm{d}v^{(0)}(\varrho(x,t))}{\mathrm{d}\varrho}. \tag{1.8}$$

Combining (1.7) and (1.8), we get the *velocity equation*:

$$\frac{\partial v(x,t)}{\partial t}+C\doteq R+A, \tag{1.9}$$

with the so-called "convection" term

$$C=v(x,t)\frac{\partial v(x,t)}{\partial x},$$

the "relaxation" term

$$R=\frac{v^{(0)}(\varrho(x,t))-v(x,t)}{\Delta t},$$

and the "anticipation" term

$$A=\frac{1}{\Delta t\cdot\varrho(x,t)}\cdot\frac{\partial\varrho(x,t)}{\partial x}\cdot\frac{\mathrm{d}v^{(0)}(\varrho(x,t))}{\mathrm{d}\varrho}.$$

We notice that the component $\frac{\mathrm{d}v^{(0)}(\varrho(x,t))}{\mathrm{d}\varrho}$ in A describes the change of the desired speed in accordance with the change of the density. The Payne model is established by the second-order partial differential equation (1.9) along with (1.2) and (1.3).

An extension of the above second-order model is the Helbing model [22]. The Helbing model takes into consideration the actual speed $v(t)$, the desired speed $v^{(0)}(t)$ of the vehicle at position $x(t)$ and introduces the notion of *time-dependent phase-space density* $\hat{\varrho}(x,v,v^{(0)},t)$ which is defined by:

$$\hat{\varrho}(x,v,v^{(0)},t)\Delta x\Delta v\Delta v^{(0)}=\frac{1}{\Delta t}\int_{t-\frac{\Delta t}{2}}^{t+\frac{\Delta t}{2}}\Delta n(x,v,v^{(0)},\tau)\,\mathrm{d}\tau,$$

where $\Delta n(x,v,v^{(0)},\tau)$ denotes the number of vehicles in the local traffic segment $\left[x-\frac{\Delta x}{2},x+\frac{\Delta x}{2}\right]$ with a current speed in the range $\left[v-\frac{\Delta v}{2},v+\frac{\Delta v}{2}\right]$ and a desired speed in the range $\left[v^{(0)}-\frac{\Delta v^{(0)}}{2},v^{(0)}+\frac{\Delta v^{(0)}}{2}\right]$ at a time $\tau\in\left[t-\frac{\Delta t}{2},t+\frac{\Delta t}{2}\right]$. The quantities Δx, Δv, $\Delta v^{(0)}$ and Δt are to be selected not only "macroscopically small" enough (as requested by the definition of integral), but also "microscopically large" enough[2] (otherwise $\Delta n(x,v,v^{(0)},\tau)$ would not be large enough to be a valid estimate of the density in the corresponding interval).

From time-dependent phase-space density the notion of *reduced phase-space density* is derived:

$$\tilde{\varrho}(x,v,t)=\int_{v^{(0)}-\frac{\Delta v^{(0)}}{2}}^{v^{(0)}+\frac{\Delta v^{(0)}}{2}}\hat{\varrho}(x,v,\nu,t)\,\mathrm{d}\nu,$$

[2]Naturally, this is not mathematically rigorous.

from which the original *spatial density* can be inferred analogously:

$$\varrho(x,t) = \int_{v-\frac{\Delta v}{2}}^{v+\frac{\Delta v}{2}} \tilde{\varrho}(x,\nu,t)\,\mathrm{d}\nu,$$

and for the average speed there is:

$$v(x,t) = \int_{v-\frac{\Delta v}{2}}^{v+\frac{\Delta v}{2}} \frac{\nu\tilde{\varrho}(x,\nu,t)}{\varrho(x,t)}\,\mathrm{d}\nu, \tag{1.10}$$

and the speed variance would be:

$$\sigma^2(x,t) = -v^2(x,t) + \int_{v-\frac{\Delta v}{2}}^{v+\frac{\Delta v}{2}} \frac{\nu^2\tilde{\varrho}(x,\nu,t)}{\varrho(x,t)}\,\mathrm{d}\nu. \tag{1.11}$$

The Helbing model, roughly speaking, is composed of definition (1.3), mass conservation (1.2), velocity equation (1.9) and a variance equation implied by (1.10) and (1.11) respecting the distribution (for example, Gaussian) of which the variable v is assumed toward its ideal value $v^{(0)}$.

1.2 Macroscopic modelling of pedestrian dynamics

In the context of pedestrian dynamics, positions and velocities are often written in two dimensions explicitly: $X = (x, y)$, $v = (v_x, v_y)$.

The conservation equation (1.1) can be rewritten as:

$$\frac{\partial\varrho}{\partial t} + \frac{\partial}{\partial x}(\varrho v_x) + \frac{\partial}{\partial y}(\varrho v_y) = 0.$$

[30] proposed a generalized model framework in which a vector field ϕ is introduced to describe the environmental influence pedestrians experience[3]. Pedestrians have the following local moving direction:

$$\begin{aligned}\hat{\phi}_x &= \frac{\frac{\partial\phi}{\partial x}}{\sqrt{\left(\frac{\partial\phi}{\partial x}\right)^2 + \left(\frac{\partial\phi}{\partial y}\right)^2}}\,,\\ \hat{\phi}_y &= \frac{\frac{\partial\phi}{\partial y}}{\sqrt{\left(\frac{\partial\phi}{\partial x}\right)^2 + \left(\frac{\partial\phi}{\partial y}\right)^2}}\,,\end{aligned} \tag{1.12}$$

[3]For the sake of a unanimous notation, here ϕ is the additive inverse of the vector field ϕ in the original text of [30]. The negative signs in (1.12) have been removed.

recalling (1.4), this in turn affects the velocity:

$$v_x = v^{(0)}(\varrho) \cdot \hat{\phi}_x,$$
$$v_y = v^{(0)}(\varrho) \cdot \hat{\phi}_y.$$

Assisted by additional assumptions, this model can be applied to describe multiple pedestrian groups.

A generalization for the case of multiple pedestrian groups is given by [31]. Let i denote the group index of the total n pedestrian groups. Regarding an individual pedestrian group, the conservation equation reads now:

$$\frac{\partial \varrho_i}{\partial t} + \nabla \cdot (\varrho_i v_i) = 0. \tag{1.13}$$

The velocity is then formulated as a sum of impacts derived from a group-specific vector field ϕ_i and an overall vector field ϕ for all groups:

$$v_i(X) = a_i \cdot v^{(0)}(\varrho_i) \cdot \phi_i(X) - b_i \left(v_{\max} - v^{(0)}(\varrho) \right) \phi(X), \tag{1.14}$$

a_i and b_i are constant control parameters; ϱ_i is the pedestrian density of group i, $\varrho = \sum_i \varrho_i$ refers to the overall density of the pedestrians; ϕ_i points to the desired moving direction of the corresponding group i, whereas ϕ stands for local corrections (owing to obstacles, high densities etc.) for all pedestrians; $v_{\max}$ stands for the maximum possible speed.

Remark: In [30], the conservation equation by pedestrian group (1.13) is formulated in the form:

$$\frac{\partial \varrho_i}{\partial t} + \nabla \cdot (\varrho_i v_i(\varrho)) = 0,$$

with $\varrho = \sum_i \varrho_i$ denoting the overall density of the pedestrians. This difference is justified by another hypothesis based on quantitative observations, see also our discussion on microscopic modelling on page 62.

A further extension is provided by [62] in which the local corrections in (1.14) are left out (that is, $b_i = 0$), but in return a Laplacian diffusion term is deployed, now the governing conservation equation (1.13) for group i has the form:

$$\frac{\partial \varrho_i}{\partial t} + \nabla \cdot \mathbf{f}_i(\varrho_0, \ldots, \varrho_{n-1}; X) = \varepsilon \nabla \cdot \nabla \varrho_i = \epsilon \Delta \varrho_i,$$

with constant control parameter ε and

$$\mathbf{f}_i(\varrho_0, \ldots, \varrho_{n-1}; X) = \varrho_i a_i v^{(0)}(\varrho) \phi_i(X)$$

is a modification[4] of the flow volume by (1.14). The flow function $\mathbf{f}_i$ can be

[4]By setting $b_i = 0$ and replacing $v^{(0)}(\varrho_i)$ with $v^{(0)}(\varrho)$, also see the previous remark for the meaning of this replacement.

re-defined to encompass a strategic component $\mathbf{f}_i^{\mathrm{s}}$ and a tactical component $\mathbf{f}_i^{\mathrm{t}}$:

$$\mathbf{f}_i = \mathbf{f}_i^{\mathrm{s}} + \mathbf{f}_i^{\mathrm{t}}.$$

The meaning of "strategic" and "tactical", in the context of pedestrian dynamics, will be explained in the next section instantly. The rather complex models of [31] and [62] are capable of reproducing certain test results close to real-world observations, however, boundary and initial conditions in the modelling partial differential equation systems require a careful calibration.

Macroscopic modelling offers a higher abstraction level, thus simulation objects will not be dealt with on an individual basis, however, it is not by coincidence that macroscopic modelling can be an adequate tool for the simulation of pedestrian (or vehicle) groups in a system. Specifically in pedestrian dynamics, many a simulation characteristic (strategic and tactical behavioural responses, for example) can be shared by both macroscopic and microscopic methods.

1.3 Microscopic modelling of pedestrian dynamics in continuous space

Microscopic modelling can be formulated in both continuous and discrete spaces. In addition, simulation objects' response to the system dynamics will be investigated on an individual basis, which is especially the case with pedestrian dynamics.

1.3.1 Behavioural patterns

Simulation of pedestrian dynamics is different from that of vehicular traffic. In general, the walking behaviour of pedestrians is complex and cannot be described solely be traffic rules.

[28] pointed out three different pedestrian behavioural types. On the *strategic* level, it is about departure, planning and activity description (sometimes called "activity pattern") of the pedestrian. However, on this stage further information of a social-psychological nature will be inevitably involved, in addition to the actual (but often unknown) circumstance signals and events. This part is assumed to be a priori knowledge in the context of the current work. The *tactical* level deals with activity scheduling, activity area choice and route choice to reach the activity area. On the lowest level, which is called *operational* level, pedestrian walking behaviour is the result of local interaction. Our local balancing mechanism (see § 3.1.3.2) offers a solution on the operational level. On the tactical level, we will suggest a new navigation method for pedestrian groups in § 3.1.5.

1.3.2 Social force model

In continuous space, the *social force* model is an important microscopic modelling ansatz [24, 25], upon which many other contributions in this field have been built. It is assumed in this model that the individual pedestrian's behavioural change is the result of various impacts, called "social forces", from other objects in the environment. The social force model simplifies the pedestrians to be *particle*s, each of these particles has the shape of a circle. Let i be the index of a certain particle, let m_i, r_i, $\vec{v}_i$ and $\vec{p}_i$ denote its mass, radius, temporary velocity and position respectively. Furthermore, we write $v_i^{(0)}$ for the desired speed and $\vec{e}_i^{(0)}$ for the local moving direction of this particle. The desired speed $v_i^{(0)}$ is a physical quantity specific to every particle i. The local moving direction $\vec{e}_i^{(0)}$ can be understood as the desired or planned moving direction on the tactical level.

At time t, the total social force regulates particle i via the following equation:

$$m_i \frac{\mathrm{d}\vec{v}_i(t)}{\mathrm{d}t} = m_i \frac{v_i^{(0)}(t)\vec{e}_i^{(0)}(t) - \vec{v}_i(t)}{\tau_i} + \sum_{j \neq i} \vec{f}_{ij} + \sum_W \vec{f}_{iW}, \tag{1.15}$$

in which $\vec{f}_{ij}$ refers to the impact from a different particle $j \neq i$ and $\vec{f}_{iW}$ is the social force imposed by a stationary object (W stands for "wall"); $\tau_i > 0$ refers to a specific time length for the measurements in the simulation, called "relaxation time" or "characteristic time".

When the distance d_{ij} between two particles i and j is larger than the sum of their radii r_i and r_j, that is, the two particles do not touch each other, the social force $\vec{f}_{ij}$ on particle i by particle j has its simplest form:

$$\vec{f}_{ij} = A_i \, e^{\frac{r_i + r_j - d_{ij}}{B_i}} \vec{n}_{ij}, \tag{1.16}$$

with positive constant parameters A_i and B_i. $\vec{n}_{ij}$ is the normalized vector pointing from particle j to particle i. In case $r_{ij} \geq d_{ij}$, additional terms to describe counter-compression and friction will be added to (1.16). The social force incurred by a stationary object $\vec{f}_{iW}$ is modelled in a similar way (omitting the counter-compression and friction terms):

$$\vec{f}_{iW} = A_i \, e^{\frac{r_i - d_{iW}}{B_i}} \vec{n}_{iW},$$

with d_{iW} being the distance from particle i to obstacle W and $\vec{n}_{iW}$ the unit vector pointing from W to the particle.

(1.15), as a first-order ordinary differential equation, can be solved by Euler's method of numerical integration. We have, respecting a given time interval length h ($0 < h \ll \tau_i$):

$$\vec{v}_i(t+h) = \vec{v}_i(t) + h \cdot \left(\frac{v_i^{(0)}(t)\vec{e}_i^{(0)}(t) - \vec{v}_i(t)}{\tau_i} + \frac{\sum_{j \neq i} \vec{f}_{ij} + \sum_W \vec{f}_{iW}}{m_i} \right), \tag{1.17}$$

which can be understood as the projection of the current velocity into a future one by knowledge of additional information respecting the environment in the system.

To simulate mass behaviour, [25] generalized the desired moving direction $\vec{e}_i^{(0)}$ of particle i into:

$$\vec{e}_i^{(0)}(t) = \frac{\vec{z}}{\|\vec{z}\|},$$

with

$$\vec{z} = (1 - q_i)\vec{e}_i^{(0)}(t) + q_i \cdot \frac{1}{n_i} \sum_{j' \neq i} \vec{e}_{j'}^{(0)}(t) \tag{1.18}$$

being the linear combination of the desired moving directions of all other particles (indexed by j') in a given neighbourhood of particle i, n_i denotes the number of such particles in this neighbourhood. The control parameter q_i in (1.18) lies in the range $[0, 1]$, obviously, $q_i = 0$ describes total individualistic behaviour and $q_i = 1$ refers to the situation of unmitigated mass behaviour.

Concerning the local position change of particle i, there is the following relationship:

$$\frac{\mathrm{d}\vec{p}_i(t)}{\mathrm{d}t} = \vec{v}_i(t),$$

numerical approximation as in (1.17) can be applied to locate the position $\vec{p}_i$ of particle i.

A modification of the social force model had been presented by [64] which assumes a linear relationship between the required length d in movement and the speed of the particle. The parameter d is then used to redefine the repulsive force (cf. (1.16)) affecting hard bodies with or without remote impact. Confirmed by empirical data, their test results showed that at higher speeds disproportionally higher space requirement of the pedestrians will be expected.

1.3.3 Position transition

Generally speaking, in each simulation model, physical magnitudes will have to be approximated at the first place. In the modelling of vehicular traffic and pedestrian dynamics this inevitably involves time and position. The system dynamics can be described collectively by the position evolution of the objects in the simulation along the time axis. For each of these, this evolution can be represented by a series of transitions of its

$$\text{position from } (x_i, y_i) \text{ to } (x_{i+1}, y_{i+1}) \text{ in time span } [t_i, t_{i+1}]$$

with $x_i, y_i, t_i \in \mathbb{R}$ $(i = 0, \ldots, T)$, $0 \le t_i < t_{i+1}$ $(i = 0, \ldots, T-1)$. The variable i is called *time index*, whereas T ($T \in \mathbb{N}^+$) is the *discrete time length*[5], denoting the total number of simulation cycles. When the context is clear, we also apply the notation $(\Delta x, \Delta y)$ to address the planar position change:

$$\Delta x = x_{i+1} - x_i, \quad \Delta y = y_{i+1} - y_i, \tag{1.19}$$

for $i = 0, \ldots, T-1$. The social force model is indeed a method for position transition on the operational level of microscopic pedestrian dynamics based on Newtonian physics. The position transition $(\Delta x, \Delta y)$ in various contexts will be discussed in the next chapters.

Remark: Although human activities take place physically in a three-dimensional space, locally speaking, they have a two-dimensional nature. The pedestrian's position change in the third dimension can be easily embedded into two-dimensional modelling components such as staircase, escalator, lift etc.

1.3.4 Cognitive modelling

[45] proposed a cognitive model for the formulation of pedestrian behaviour with the help of additional visual information. It is assumed that the pedestrian chooses a walking direction which would allow the most direct passage to the destination while taking into consideration the presence of all obstacles in the environment.

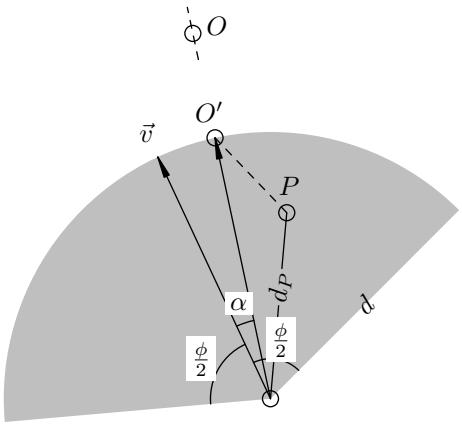

Figure 1.1: Illustration of a perception area in cognitive modelling. O is the planned destination, O' is the intersection point of this main direction with the bounding arc of the current perception area, d_P is the distance to the next pedestrian expected as obstacle at position P.

[5] An often neglected fact—though in most cases unknowingly assumed—is that it is not possible for an electronic machine to store and process real numbers with *infinite* precision. Hence in the simulation, time can always be considered as a discrete variable.

Let O denote the planned destination. Given a range of perception d, a previous moving direction $\vec{v}$, the *perception area* of a pedestrian would be the circular sector with radius d and angle ϕ. The new walking direction, expressed as angular deviation β from the previous walking direction $\vec{v}$, should minimize the following term:

$$d^2 + f^2(\beta) - 2d \cdot f(\beta) \cos(\alpha - \beta), \tag{1.20}$$

in which α is the angular deviation from the previous direction $\vec{v}$ to the direction pointing toward destination O, f is a function of β, to be set either as d, if there is no obstacle in that direction, or the distance to the first obstacle. An illustration is given in Figure 1.1.

To be more specific, if in a certain direction no obstacle is expected, (1.20) is minimized by setting $\beta = \alpha$, that is to say, a direction correction pointing to the destination O is to be undertaken. Otherwise, if β points to an expected obstacle, for example, P in Figure 1.1, (1.20) becomes the square of the distance $O'P$, minimization of (1.20) leads to the finding of an intermediate position with which the remaining distance to O is minimized as well. Solving β in (1.20) can be a computationally expensive task, since all obstacles in the perception area with their potential position transitions must be taken into consideration, and β is to be solved for every pedestrian. Discussion of further models based on obstacle avoidance can be found in [39].

The cognitive model [45] offers a solution on the tactical level of pedestrian dynamics and can be combined with the social force model [24, 25]. We will encounter the notion of perception area in § 3.1.6.2 again.

Chapter 2

Grid-based methods for vehicular traffic

A special category of the microscopic methods is the cellular automata (CA) model which has proven to be a useful tool for the simulation of vehicular traffic and pedestrian dynamics.

2.1 Cellular automata

The notion of cellular automaton (CA) finds its genesis in the study of self-reproduction from a mathematical perspective [49] by John von Neumann in the late 1940s. The result was a hypothetical system with a set of very complicated rules defined on a rectangular grid. Two decades later, a computer game named "Game of Life" devised by the British mathematician John Horton Conway aroused much interest and attention [4]. Strictly speaking, "Game of Life" is not a game—since it is not interactive—, it is about the evolution of the system states, starting with a given configuration provided by the player, under the application of a few simple rules. The "Game of Life" showed in a way more than astounding the various patterns of evolution and self-organization.

In 2002, Stephen Wolfram published his complex but not uncontroversial (see the comments in [20], for example) volume [73]. Unlike Conway's approach of first defining a two-dimensional automaton (with simple but carefully chosen rules) and then studying the behaviour of system evolution, [73] followed the line of enumerative experiments of one-dimensional automata via computers and classified these vaguely into four categories, according to the system behaviour they exhibit. It had been conjectured that one category of these CA is computationally universal (Turing-complete).

2.1.1 Formal definition of CA

A cellular automaton is composed of a discrete lattice of cells of a universal shape, a set of states of the cells, a neighbourhood relationship defined on the cells and a transition function defined on the cell and its neighbourhood. The transition

function can be considered as a set of transition rules, especially when the function is discrete itself. The discrete lattice is sometimes also called "universe" and is not confined to be two-dimensional. This lattice of cells will be referred to as the physical environment or the geometry of the CA.

Definition. *(after [35], with slight notational modifications) An n-dimensional CA is a tuple (L^n, S, N, δ), where L^n is the index set of cells in dimension n, S is the set of cell states, $N = (\vec{x}_1 \ldots, \vec{x}_{|N|})$, the neighbourhood relationship, is a vector of pairwise different elements from L^n, and $\delta : S^{|N|} \to S$ is the transition function. The elements of N will be called neighbourhood patterns. When the CA is spatially infinite, L can be written as $\mathbb{Z}$ (or $\mathbb{N}$) explicitly.*

Let us have a look of the simplest case $n = 1$.

Example 2.1. *With $n = 1$, the vector element $\vec{x} \in L^n$ now degenerates into a scalar in L. In this case, a CA is called elementary if it has exactly two cell states*[1] *and a neighbourhood relationship of radius 1. This can be written as $S = \{\mathbf{0}, \mathbf{1}\}$ and $N = (-1, 0, 1)$. We notice that there are $|S|^{|N|} = 2^3 = 8$ different neighbourhood patterns. Since $|S| = 2$ and $|N| = 3$, there are $|S|^{|S|^{|N|}} = 2^8 = 256$ different elementary cellular automata in all. For simplicity's sake, the transition function $\delta : \{\mathbf{0}, \mathbf{1}\}^3 \to \{\mathbf{0}, \mathbf{1}\}$ can now adopt the following form:*

$$\begin{array}{ccc} \mathbf{l} & \mathbf{c} & \mathbf{r} \\ & \mathbf{s} & \end{array} \tag{2.1}$$

In (2.1), $\mathbf{l}$ ("left"), $\mathbf{c}$ ("centre") and $\mathbf{r}$ ("right") represent the states in three connecting cells which further correspond to a neighbourhood pattern. The new state in the centre cell after transition is denoted by $\mathbf{s}$. When a CA is considered as a system with its own dynamics defined by its transition function δ, the evaluation of the latter can be naturally seen as the update of the system. All eight possible neighbourhood patterns for (2.1), each of which associated with a new cell state, can be enumerated:

$$\begin{array}{cccccccccccc} \mathbf{1} & \mathbf{1} & \mathbf{1} & & \mathbf{1} & \mathbf{1} & \mathbf{0} & & \mathbf{1} & \mathbf{0} & \mathbf{1} & & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ & \mathbf{s_7} & & & & \mathbf{s_6} & & & & \mathbf{s_5} & & & & \mathbf{s_4} & \\ \mathbf{0} & \mathbf{1} & \mathbf{1} & & \mathbf{0} & \mathbf{1} & \mathbf{0} & & \mathbf{0} & \mathbf{0} & \mathbf{1} & & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ & \mathbf{s_3} & & & & \mathbf{s_2} & & & & \mathbf{s_1} & & & & \mathbf{s_0} & \end{array} \tag{2.2}$$

In this way, the transition function δ can be unambiguously expressed by the combination of eight new cell states $\mathbf{s_7}, \ldots, \mathbf{s_0}$. Since by definition $\mathbf{s}_i \in S = \{\mathbf{0}, \mathbf{1}\}$ ($i = \mathbf{0}, \ldots, \mathbf{7}$), this combination of cell states can be further compressed into a binary string $(\mathbf{s_7}\,\mathbf{s_6}\,\mathbf{s_5}\,\mathbf{s_4}\,\mathbf{s_3}\,\mathbf{s_2}\,\mathbf{s_1}\,\mathbf{s_0})_2$ and interpreted as an integer number in the range $0, \ldots, 255$. A naming scheme suggested by [73] is to address the 256

[1] In comparison, the self-reproducing automaton described in [49] contains twenty-nine states.

elementary cellular automata by these numbers.

2.1.2 CA in first application

Many discrete traffic models start with one dimension and two cell states. The state **1** is used to indicate that a simulation object is present in the current cell; logically, **0** stands for the "free" state which indicates that entering into the current cell for other simulation objects is possible. We assume, without loss of generality, a flow direction in the traffic system from left to right (i. e. in the positive order of L).

Example 2.2. *The famous Rule 184 refers to the following elementary CA:*

$$\begin{array}{cccccccccccccc} \mathbf{1} & \mathbf{1} & \mathbf{1} & & \mathbf{1} & \mathbf{1} & \mathbf{0} & & \mathbf{1} & \mathbf{0} & \mathbf{1} & & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ & \mathbf{1} & & & & \mathbf{0} & & & & \mathbf{1} & & & & \mathbf{1} & \\ \\ \mathbf{0} & \mathbf{1} & \mathbf{1} & & \mathbf{0} & \mathbf{1} & \mathbf{0} & & \mathbf{0} & \mathbf{0} & \mathbf{1} & & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ & \mathbf{1} & & & & \mathbf{0} & & & & \mathbf{0} & & & & \mathbf{0} & \end{array} \tag{2.3}$$

since it can be encoded as $(\mathbf{10111000})_2 = 184$ *with the notation of* (2.2).

An elementary CA implies that its neighbourhood is of radius 1 *(cf. Example 2.1). Hence,* (2.3) *describes a system in which the maximum speed is* 1 *(measured in number of cells per update, often written as* $v_{\max} = 1$*). It is clear that a cell in current state* **1** *retains this state if and only if its immediate neighbouring cell on the right side has the state* **1***, that is, a forward movement for the object in the current cell is denied. In analogy, a cell in current state* **0** *retains this state if and only if its immediate neighbouring cell on the left side has the state* **0***, which means that there is no follower present to claim the current cell after the update.*

If the states **0** and **1** have no effect in a transition rule, they can be replaced by a meta-state "−". In such a way, (2.3) can be rewritten as

$$\begin{array}{cccccccccccccc} - & \mathbf{1} & \mathbf{1} & & - & \mathbf{1} & \mathbf{0} & & \mathbf{1} & \mathbf{0} & - & & \mathbf{0} & \mathbf{0} & - \\ & \mathbf{1} & & & & \mathbf{0} & & & & \mathbf{1} & & & & \mathbf{0} & \end{array}$$

this meta-state "−" is sometimes called "don't care" state as well.

Technical note. In fact, the cell state set S can contain elements other than **1** and **0**. In a broader sense, S does not need to be finite, it can be $\mathbb{Z}$ or even $\mathbb{R}$. In the latter case, the transition can be seen as a real-valued function. Such a CA model is sometimes called *coupled map*, which belongs to a special category of dynamical systems defined on discrete time and spaces (see for example, [18]). Furthermore, the transition function δ may be *stochastic*, that is, the transition rule to be applied on the cells can be performed on a probabilistic basis. CA models without any stochastic element is called *deterministic*.

We take a second look of the neighbourhood relationship N. In the elementary case of Example 2.1, we define the radius of N to be 1, this implies that for a simulation object in the system, the position transition (in one dimension, since the cellular automata referred to are elementary) after an update is limited to a maximum length of 1, so the maximum locally achievable speed is $v_{\max} = 1$, cf. Example 2.2. Obviously, the transition function describing $v_{\max}$ ($v_{\max} \in \mathbb{N}^+$) requests a neighbourhood of radius r not smaller than $v_{\max}$. Given a nontrivial radius r of the neighbourhood, a position transition of r cells can be carried out if all the r cells in the corresponding direction are in state $\mathbf{0}$. This is to say, a successful forward step of length r at a location in state $\mathbf{1}$ takes place if and only if

$$\begin{array}{cccc} & & \mathbf{1} & \mathbf{0}^r \\ & & \mathbf{0} & \end{array}^{-r}$$

where $\mathbf{s}^r$ denotes the exact r-times repetition of a state $\mathbf{s}$; whereas a forward move with a local speed r to a location in state $\mathbf{0}$ is to be carried out if and only if

$$\begin{array}{cccc} \mathbf{1} & \mathbf{0}^{r-1} & \mathbf{0} & \\ & & \mathbf{1} & \end{array}^{-r}$$

i. e. the follower is in a correct position and there is no hindrance for this position transition.

We recognize that there are 2^{2r+1} neighbourhood patterns applicable in the transition function δ. For $r = 2$, after a somehow tedious enumeration, the transition function can be encoded as $(\mathbf{11100000111011111110000011100000})_2$, starting with the neighbourhood pattern $\begin{array}{ccccc} \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ & & \mathbf{1} & & \end{array}$ as the highest bit and descending to the lowest bit $\begin{array}{ccccc} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ & & \mathbf{0} & & \end{array}$.

2.2 CA in vehicular traffic simulation

We recall that the transition function δ of a CA projects the states in the neighbourhood of a certain position in L (which has been called "cell" as well) into a new state at this position. The overall state transition can be used to simulate vehicular traffic systems. From a practical perspective, once the state transition—which is formally to be understood as the result of the application of pre-defined transition rules—is clear, there will be no real need for a formal definition of the CA, if we are interested in the system behaviour only.

2.2.1 Stochastic model by Nagel and Schreckenberg

[47] has been the first application of CA for the simulation of vehicular traffic. It is a single-lane model with $v_{\max} > 1$. Discrete positions in the traffic lane can be modelled by the cells of a CA. The system dynamics can be completely described

by the state transition (and evolution) of all the cells in the system. The state transition of the cell is subject to a set of simple rules.

Let x and v denote position and speed (in discrete sense) respectively. The transition function of this model can be translated into the following rules expressed in x and v:

$$v \Leftarrow \min(v+1, v_{\max}), \tag{2.4a}$$

$$v \Leftarrow \min(x' - x - 1, v), \tag{2.4b}$$

$$v \Leftarrow \max(0, v-1), \text{executed with a given probability}, \tag{2.5}$$

$$x \Leftarrow x + v. \tag{2.6}$$

The superscript $'$, as seen in (2.4b), refers to the immediate neighbour in the flow direction of the simulation object being studied. x' denotes the position of the immediate leading vehicle in the flow direction. (2.4a) describes the vehicle's acceleration, whenever it is technically possible. (2.4b) tells how fast the vehicle is allowed to be at the moment. Since x' refers to the position of leading vehicle, $x' - x - 1$ will be the maximum[2] space the current vehicle is allowed to cover in the upcoming update free of collision in the flow direction. This is sometimes referred to as the rule of *forward causality*. (2.5) can be considered as a random deceleration (braking-down) of the vehicle which is also a real-world phenomenon, since drivers sometimes slow down their vehicles for no particular reason. Carrying out this step on a random basis leads the model to be nondeterministic. Finally, the position of the vehicle will be updated by (2.6). In every simulation cycle, the system update runs in parallel for all vehicles and, for each of these, the four consecutive steps (2.4a)–(2.6) are to be carried out.

Remark: The formulation of (2.4a) and (2.4b) is slightly different from the original definition in [47], which reads:

> 1) Acceleration: if the velocity v of a vehicle is lower than $v_{\max}$ and if the distance to the next car ahead is larger than $v+1$, the speed is advanced by one $[v \to v+1]$.
> 2) Slowing down (due to other cars): if a vehicle at site i sees the next vehicle at site $i+j$ (with $j \leq v$), it reduces its speed to $j-1$ $[v \to j-1]$.

In the original text, j refers to $x' - x$ in our notation. Although the semantic of 1) and 2) is not identical with that of (2.4a) and (2.4b), the algorithmic result is the same.

Proof. Consider $v = v_{\max}$. By (2.4a), we have $v \Leftarrow v_{\max}$; in comparison, 1) is not applicable, so $v = v_{\max}$ stays unchanged. Further, if $x' - x \leq v_{\max}$, by (2.4b), $v \Leftarrow x' - x - 1$; by 2), since $j = x' - x$, we have $v \Leftarrow j - 1$, this delivers the same result as (2.4b). If otherwise $x' - x > v_{\max}$, by (2.4b), $v \Leftarrow v$ (since $x' - x - 1 \geq v_{\max}$ and $v = v_{\max}$), whereas 2) is not applicable, so $v = v_{\max}$ remains again unchanged.

[2] At this moment, the state of the vehicle ahead after the update is unknown, so its position should be unavailable during the current update. In reality, however, a position transition of $x' - x$ will not inevitably cause a collision, since the leading vehicle may have moved to a new position ahead.

Consider otherwise $v < v_{\max}$. We set $v_{\text{temp}} \Leftarrow v + 1$ by (2.4a). To evaluate 1), we need to know $x' - x$. If $x' - x < v_{\text{temp}}$, we have $v \Leftarrow x' - x - 1$ by (2.4b). On the other hand, $x' - x < v_{\text{temp}}$ implies $x' - x < v + 1$, that is, $x' - x \leq v$, therefore, noticing 1) is not applicable, by 2) we have the same result $v \Leftarrow x' - x - 1$. If $x' - x > v_{\text{temp}}$, we have $v \Leftarrow v_{\text{temp}}$ by (2.4b), that is, $v \Leftarrow v + 1$. For the same $x' - x > v_{\text{temp}}$, 1) is applicable, we have the same result to increment v, and 2) is not applicable after this update. Finally, if $x' - x = v_{\text{temp}}$ (that is, $x' - x = v + 1$), we have $v \Leftarrow v$ by (2.4b), v stays unchanged, the same is with the original text, since neither 1) nor 2) is applicable. □

The advantage of our revision is that (2.4a) and (2.4b) can now be condensed into a single step:

$$v \Leftarrow \min(x' - x - 1, v + 1, v_{\max}).$$

In comparison, 1) and 2) in the original text refer to two operations with compensating effects (that is, acceleration and deceleration) as two consecutive steps, this may appear to be less efficient in the modelling.

In the original model, each discrete position in the lane has a length of 7.5 m, with a time interval length of one second in ($\Delta t = 1\,\text{s}$) the simulation cycle and $v_{\max} = 5$, this would induce a maximum speed of $135\,\text{km} \cdot \text{h}^{-1}$ for the vehicles.

Remark: In discrete form, speed v will be measured in positions of the underlying geometry. A discrete speed v, which is dimensionless, refers to the position transition of v positions in one simulation cycle Δt.

2.2.2 A two-lane extension

Vehicles in reality have different speed capacities, when the model in § 2.2.1 is applied on these, it would result inevitably in the so-called undesired "platooning"[3] phenomenon in that the free flow of the faster vehicles is hindered by the slower ones. [58] introduced an extended model which consists of two independent single lanes. The update strategy in the simulation cycle is composed of two substeps. The first one is to make decisions about possible lane changing while the second executes the corresponding position transitions on both lanes according to the result of the first substep. Performing independent position transitions on both lanes means that each of the two lanes will be updated as if it were in the single-lane model [47].

The two lanes will be labelled by x and y. Applying our notation, respecting a certain vehicle at position x' in the x-lane, x'' refers to the position of the immediate leading vehicle in the same lane x and y'' refers to the position of the immediate leading vehicle in the y-lane; and similarly, y denotes the position of its immediate follower there. The first substep can be translated into the

[3]The otherwise desired "platooning" refers to the method of grouping vehicles for a better utilization of road resources and higher fuel efficiency by means of reducing distance among the former.

following rule set:

$$x'' - x' - 1 < l_x, \tag{2.7}$$

$$y'' - x' - 1 > l_y, \tag{2.8}$$

$$x' - y - 1 > l_y^-. \tag{2.9}$$

l_x, l_y and l_y^- are system parameters. For the vehicle at position x', changing into lane y is possible, when the conditions (2.7), (2.8) and (2.9) are fulfilled; this lane changing will be carried out with a certain given probability p (cf. (2.5)) when it becomes possible.

Remark: There exist both right-hand and left-hand traffic systems. In the sequel, the right side in the right-hand traffic system as well as the left side in the left-hand system will be called the "inner" side, the left side in the right-hand system and right side in the left-hand system will be called the "outer" side.

(2.7) says that if the space in front of the vehicle at position x' is smaller than a particular value l_x, the driver of the vehicle would seek a chance to change lane. If this rule is also set as a condition for vehicles in the other lane, the model will be called *symmetric*. As a second condition, (2.8) requests sufficient space l_y in the lane to change into for a possible lane changing. The original model suggested $l_y = l_x$. Similar to (2.4b), (2.7) and (2.8) define the rule of *forward causality*. In analogy, (2.9) stands for the so-called *backward causality*. A positive l_y^- requests that a lane changing is only possible when there is enough (real) space for the follower in the lane to change into. Setting $l_y^- = 0$ (or $l_y^- < 0$) means that backward causality is not requested.[4]

Remark: A second thought must be given to the so-called "symmetry" of the lanes. In Germany as well as many other European countries, driving on the inner lane (that is, the right lane in the right-hand traffic system) is mandatory and overtaking from the inner lane is generally forbidden. In such a case, lane usage is inhomogeneous and lane changing is to be considered as an *asymmetric* process, since vehicles are requested to stay in the inner lane unless when overtaking other vehicles.

Apart from $l_y = l_x$ and the probability p which takes control of the actual execution of a possible lane changing, [58] experimented with various parameter combinations: $l_x = v + 1$ or $l_x = v$, $l_y^- = 0$ (without backward causality) or $l_y^- = v_{\max}$ (maximum backward causality). Whereas $l_x = v$ is not too much different from $l_x = v + 1$ at higher speeds, in the case of $v = 0$ (that is, traffic

[4] By the definition of "gap" in the original text and the claim that backward causality is dismissed by $l_y^- = 0$, it seems that (2.9) needs to be relaxed into

$$x' - y - 1 \geq l_y^-.$$

Similarly, (2.8) should be modified into

$$y'' - x' - 1 \geq l_y.$$

jam), the system behaviour is very different, since the condition (2.7) cannot be fulfilled (by $l_x = v$, that is, $l_x = 0$), a lane changing will not be possible even if the neighbouring lane is completely free; the current vehicle (which is being stopped) will have to wait until the leading vehicle in this lane moves forward. It is also interesting to observe that in this two-lane traffic system, $l_y^- = 0$ "corresponds to a very egoistic driver behaviour", since "vehicles no longer check whether their lane changing could have a disadvantageous effect on the other lane".

Remark: The position update of the vehicles in the second substep takes place in a parallel manner. To our knowledge, there is no straightforward way to further extend this model to suit the generic case of multiple lanes, since the decision about potential lane changing can be affected by vehicles from bordering lanes of both sides. To solve this problem without technical compromise, more advanced technique involving *semaphore* (introduced by [15]) will be necessary.

2.3 A deductive multi-lane extension

Safety distance is another issue not to be neglected. In Germany, a minimum safety distance in metres as the half of vehicle speed measured in kilometres per hour is recommended and, in most cases, obligatory. With consideration of the dynamic safety distance in this aspect, we construct a model for vehicular traffic with multiple lanes in discrete space. This model can be considered as *deductive* by definition of [27], since system parameters are derived from known physical quantities. The first part of the model had been explained in our previous work [12].

2.3.1 Parameter deduction

We take a closer look of the real distance between the vehicles. Let the length of an average vehicle be l_v. Let the lanes be equally divided into cells with a length of l_c. The collision-free distance of two vehicles positioned at x and $x + d$ would be $d \cdot l_c - l_v$, see Figure 2.1. A very realistic estimate, according to [47], would be $l_v = 5\,\mathrm{m}$. Under this assumption, a discrete position length of $l_c = 15\,\mathrm{m}$ gives a distance of 10, 25, 40, 55 and 70 metres for $d = 1, 2, 3, 4, 5$ respectively, which further refers to a local speed limit of 20, 50, 80, 110 and $140\,\mathrm{km} \cdot \mathrm{h}^{-1}$ respectively. In other words, at $50\,\mathrm{km} \cdot \mathrm{h}^{-1}$, for instance, safety distance of an additional empty discrete position (with a length of 15 m) on the lane must be respected. In the model of [47], however, the distance of vehicles at two adjacent positions is $7.5\,\mathrm{m} - l_v$, which is much too small from a practical perspective.

In the model of [47], if the time length of the simulation cycle Δt is set to be one second, change of one discrete position in the lane would refer to a temporary speed of $27\,\mathrm{km} \cdot \mathrm{h}^{-1}$. In our model the position length in the lane has been doubled ($l_c = 15\,\mathrm{m}$), and we set the time length of the simulation cycle Δt

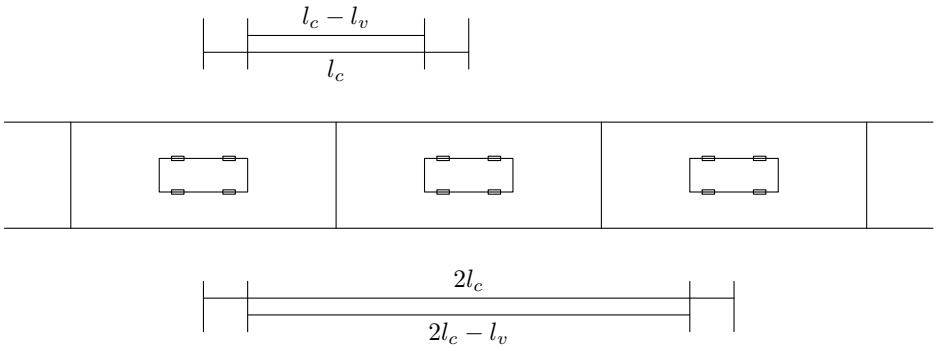

Figure 2.1: Physical distance between two passenger cars in a traffic lane. The distance between two vehicles at two adjacent positions (drawn as enclosing rectangles) is $l_c - l_v$, instead of l_c or 0, as often seen in other discrete models; the collision-free distance of a vehicle to another one at a position d cells away is $d \cdot l_c - l_v$.

to be 2.4 s. This has the advantage that change of one discrete position in the lane refers to a temporary speed of $22.5\,\mathrm{km \cdot h^{-1}}$; in addition, the acceleration of one discrete position in the simulation cycle would be equal to a speed change of $9.375\,\mathrm{km \cdot h^{-1}}$ every second (and consequently, about eleven seconds from 0 to $100\,\mathrm{km \cdot h^{-1}}$) which resembles the capacity of an average passenger car nowadays. $v_{\max} = 6$ would correspond to a maximum speed of $135\,\mathrm{km \cdot h^{-1}}$ in our model.

2.3.2 Lane deployment and notation

In generic multiple lane situations, there are only three types of lanes to consider: the two border lanes in which lane changing is allowed only in one direction and the middle lane in which lane changing in both directions are possible. Without loss of generality, the lanes will be written as L with an index $l = 0, 1, 2$ (counted from the innermost lane). Adding further lanes in the middle should be straightforward in the implementation. A possible lane changing can be denoted by $\Delta l \in \{-1, 0, 1\}$, when this operation is well-defined in the system. To simplify our notation, positions on the lanes $l = 0, 1, 2$ will be written in x, y and z respectively. As in the previous text (see page 15), the immediate leading vehicle in the flow direction will be addressed by the superscript $'$: for a vehicle at position x, the vehicle directly ahead of it will be at position x'; by x' and z' we mean the positions of the vehicles ahead of a vehicle positioned at y in the corresponding lanes ($l = 0$ and $l = 2$). Furthermore, the speed of the vehicle at position x will be written as v_x.

2.3.3 Safety distance

We introduce a safety distance coefficient c ($c \geq 1$) with which, in discrete form, the safety distance respecting a speed v ($v = 0, \ldots, v_{\max}$) can be defined[5]:

$$s_{c,v} = \frac{\frac{c}{2} \cdot v \cdot l_c \cdot \frac{1\,\mathrm{h}}{\Delta t} \cdot 10^{-3}}{l_c}. \tag{2.10}$$

We notice that $s_{c,v}$ is independent of l_c. With $\Delta t = 2.4\,\mathrm{s}$, (2.10) becomes

$$s_{c,v} = \frac{3cv}{4}. \tag{2.11}$$

Given a discrete distance d to the immediate leading vehicle, a collision-free speed v (in discrete form) is bounded by

$$v \cdot 1 + s_{c,v} \leq \frac{d \cdot l_c - l_v}{l_c} = d - \frac{l_v}{l_c}. \tag{2.12}$$

In (2.11), $c = 1$ corresponds to the minimum safety distance imposed by traffic regulations which leads to $s_{1,v} = \frac{3}{4}v$ and $v \leq \frac{4}{7} \cdot \left(d - \frac{l_v}{l_c}\right)$. When c is clear in the context, $s_{c,v}$ can be shortened as s_v.

2.3.4 Driving strategy

In Germany as well as many other countries, driving on the inner lane (that is, the right lane in the right-hand traffic system) is mandatory and overtaking from the inner lane is prohibited. Vehicles are generally requested to keep on the inner lane unless when overtaking other vehicles. Consequently, the inner lanes are associated with lower vehicle speeds. In such a case, lane usage is inhomogeneous and lane changing is considered to be an *asymmetric* process.[6]

2.3.4.1 Asymmetric case

Same as in the single-lane model, local speed of the vehicle will be increased whenever possible, see (2.4a). After this, decision about position change and possible lane changing is to be made in the simulation cycle for every vehicle. This process contains the following four components.

[5] In (2.10), the term $\frac{1\,\mathrm{h}}{\Delta t}$ gives the number of the simulation cycles in one hour; $v \cdot \frac{l_c}{1\,\mathrm{m}} \cdot \frac{1\,\mathrm{h}}{\Delta t} \cdot 10^{-3}$ gives the dimensionless speed quantity measured in $\mathrm{km} \cdot \mathrm{h}^{-1}$ associated with the current discrete speed v. The conversion of quantity in physical speed into physical length by the traffic rule respecting safety distance introduces implicitly a multiplier which will counterbalance the denominator 1 m.

[6] In comparison, in North America the traffic system is symmetric, in most cases overtaking is allowed on both sides.

2.3.4.1.1 Check inner lane The vehicle at position y' seeks the chance to change into the inner lane, as required in an asymmetric traffic system. Obviously, such a driving manoeuvre is only possible when $l > 0$. With

$$x'' - y' - 1 - s_{v_{y'}} \geq v_{y'}, \tag{2.13a}$$

and

$$v_{y'} > v_x, \tag{2.13b}$$

a lane changing $\Delta l = -1$ will be possible for the current vehicle at a speed $v_{y'}$. (2.13a) refers to the forward causality mentioned in (2.8). The backward causality is now replaced by a straightforward speed in comparison with the immediate following vehicle of the inner lane (2.13b): a higher speed of the current vehicle would guarantee a collision-free lane changing. This modification eliminates the usage of the parameter l_y^- in (2.9).

2.3.4.1.2 Check current lane The vehicle inspects the situation in the current lane. With

$$y'' - y' - 1 - s_{v_{y'}} \geq v_{y'}, \tag{2.14}$$

no lane changing will be necessary ($\Delta l = 0$) and the current speed $v_{y'}$ will be maintained. Since no lane changing is involved, backward causality (such as $v_{y'} > v_y$ or $v_{y'} \geq v_y$) needs not to be considered,[7] forward causality (2.14) becomes the only question. We notice that (2.14) is easier than (2.13a) to meet, since in outer lanes the vehicles usually have higher speeds and consequently the density should be lower which further leads to $y'' - y' > x'' - y'$ statistically. This explains why vehicles have no need to change lanes too often.

2.3.4.1.3 Check outer lane Now the vehicle under study—still at position y', since the last two operations have been without success—attempts to change into the outer lane. To perform this operation, there must be $l < 2$ and

$$z'' - y' - 1 - s_{v_{y'}} \geq v_{y'}, \tag{2.15a}$$

and

$$y' - z \geq v_{\max}. \tag{2.15b}$$

In such a case, a lane changing $\Delta l = 1$ will be possible at the current speed $v_{y'}$. In addition to the forward causality expressed in (2.15a), backward causality is now recovered in (2.15b). We do not request $v_{y'} > v_z$, because a reliable estimate of the speed becomes difficult with the increasing vehicle speed in the outer lane (in an asymmetric traffic system). Instead, $v_{\max}$ is adopted to ensure sufficient

[7] Backward causality in the current lane can be considered as the problem of forward causality for the following vehicle in this lane.

safety.

2.3.4.1.4 Move in current lane In the current lane, by (2.12), the maximum local speed with consideration of safety distance can be deduced:

$$v \Leftarrow \min\left(\frac{4}{4+3c}\cdot\left(d-\frac{l_v}{l_c}\right), v\right),$$

here $d = y'' - y'$ denotes the position difference with the leading vehicle. The value of v is in general not an integer.

In addition, we need to know whether the space $y'' - y' - 1$ is sufficient for the current vehicle to move forward in the flow direction. To this end, we propose:

$$v \Leftarrow \begin{cases} y'' - y' - 1, & \text{if } v \geq y'' - y' - 1, \\ v^*, & \text{otherwise,} \end{cases}$$

with

$$v^* = \begin{cases} \lfloor v \rfloor + 1, & \text{if } p < v - \lfloor v \rfloor, \\ \lfloor v \rfloor, & \text{otherwise,} \end{cases}$$

where p is a random number from $[0, 1)$ and $\lfloor \cdot \rfloor$ refers to the largest integer no greater than the argument. The construction of v^* ensures that the position update of the vehicle can be performed accurately. With $\Delta l = 0$, the vehicle moves forward in the flow direction at the speed v.

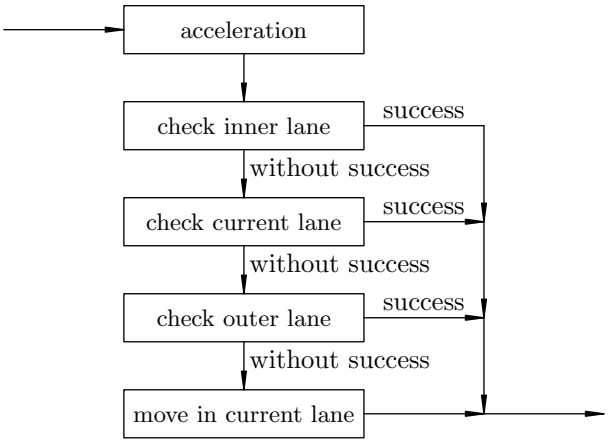

Figure 2.2: Flow chart of the asymmetric driving strategy for individual vehicles in a simulation cycle.

To summarize the asymmetric case, we start with § 2.3.4.1.1, if changing into the inner lane cannot be performed, we switch to §2.3.4.1.2; if (2.14) is not fulfilled, we check operation in the outer lane §2.3.4.1.3; if not both (2.15a)

and (2.15b) are satisfied, the vehicle will have to stay in the current lane §2.3.4.1.4. The overall driving strategy of the asymmetric case is illustrated in Figure 2.2.

2.3.4.2 Symmetric case

Symmetric traffic systems are typically seen in North America. In such a system, the vehicles are requested—partly owing to the heavy traffic loads—to stay in their lanes whenever possible. The overall traffic system appears to be more homogeneous (in terms of vehicle density and speed) than in the asymmetric case. On the other hand, when a lane changing is feasible, overtaking is allowed on both sides.

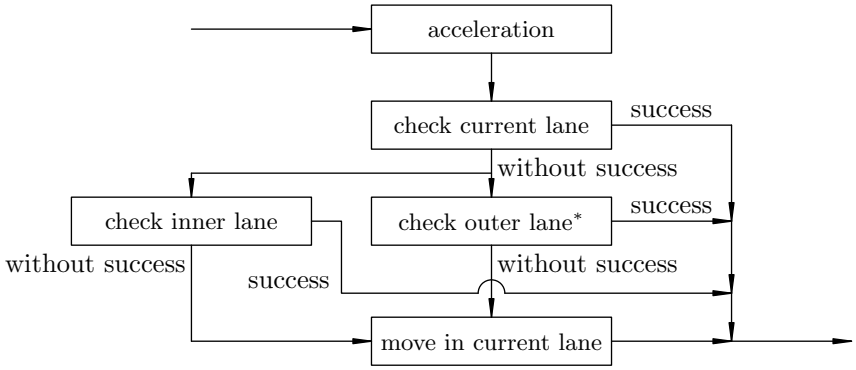

Figure 2.3: Flow chart of the symmetric driving strategy for individual vehicles in a simulation cycle. The choice between "check inner lane" (§ 2.3.4.1.1) and "check outer lane*" (referring to the modification defined by (2.15a) and (2.16)) is to be made on an equal basis.

Lane changing in the symmetric case should be given consideration on an equal basis; apart from this, another significant change concerns the backward causality (2.15b) in § 2.3.4.1.3 ("check outer lane"), which we revise into:

$$v_{y'} > v_z. \tag{2.16}$$

With this modification (cf. (2.13b)), symmetric behaviour of the system can be expected. Figure 2.3 illustrates the overall driving strategy in the symmetric case.

Remark: As mentioned earlier (see page 18), there is no straightforward way to apply parallel update on more than two lanes, we would prefer a random sequential state update of the vehicles; technically, it is possible to disintegrate the complete lanes into lane blocks and apply state update on these blocks simultaneously to achieve a better parallelism of the whole simulation system.

Technical note. In § 2.3.2 we introduced the superscript $'$ to address the leading position of a vehicle in multiple lanes. Respecting a given position y', for the position of the immediate leading vehicle in the same lane y'' there is $y'' > y'$. In the other lanes, however, we request $x'' \geq y'$, $z'' \geq y'$. For $x'' - y' = 0$ and $z'' - y' = 0$, the respecting forward causality in (2.13a) and (2.15a)) will not be fulfilled, this confirms the fact that no lane changing should take place when two vehicles are at a same position in two neighouring lanes. In analogy, concerning the followers of the vehicle at position y', we request $x \leq y'$, $y < y'$ and $z \leq y'$. In this case, asymmetric backward causality (2.15b) will be deactivated for two vehicles at a same position in two neighouring lanes ($z = y'$, $v_{\max} > 0$).

2.3.5 Simulation results

For the tests of our model with various parameter configurations, periodic boundary of the lanes has been applied. This means that all vehicles were placed in a closed circulating traffic system. Naturally, such a closed system can be integrated into larger, open systems with ease. Owing to this periodic boundary, the actual length of the lanes became insignificant for the simulation. The vehicles were assigned a default discrete speed of 3 (that is, $67.5\,\mathrm{km}\cdot\mathrm{h}^{-1}$). Because of the automatic acceleration (2.4a), this default initial speed value had no substantial impact on the overall dynamics of the system. The maximum discrete speed was set to be $v_{\max} = 6$ ($135\,\mathrm{km}\cdot\mathrm{h}^{-1}$). The system dynamics can be studied on two different levels.

2.3.5.1 Space-time diagram

We first study the so-called space-time relationship. The so-called *space-time diagram*s records the position evolution of the vehicles in a traffic system. A space-time diagram is sometimes called vehicle trajectory plot as well.

Experiments in this section have been conducted on four lanes with different parameter settings. The initial vehicle density of lane i is denoted by p_i ($i = 0, 1, 2, 3$). The state evolution of the lanes is presented in form of the traces of the individual vehicles in the corresponding lanes. An example of the space-time diagram is given in Figure 2.4. The traces of the individual vehicles are shown in random colours. Due to the periodic condition, the trace of a vehicle is discontinued when crossing the boundary. The trace of a vehicle is also discontinued in case of a lane changing. When a vehicle changes into the current lane (of the trajectory plot), the corresponding position change is shown in a small dashed line segment (please zoom in on the electronic version of this text to see the details). Obviously, the maximum density 1 is equal to the inverse of the length of each site (discrete position) in the lane, that is, $\frac{1}{l_c}$, which further refers to approximately 67 vehicles per kilometre. Average speed is measured only when the corresponding lane is not empty. Figures 2.5 to 2.10 present the space-time

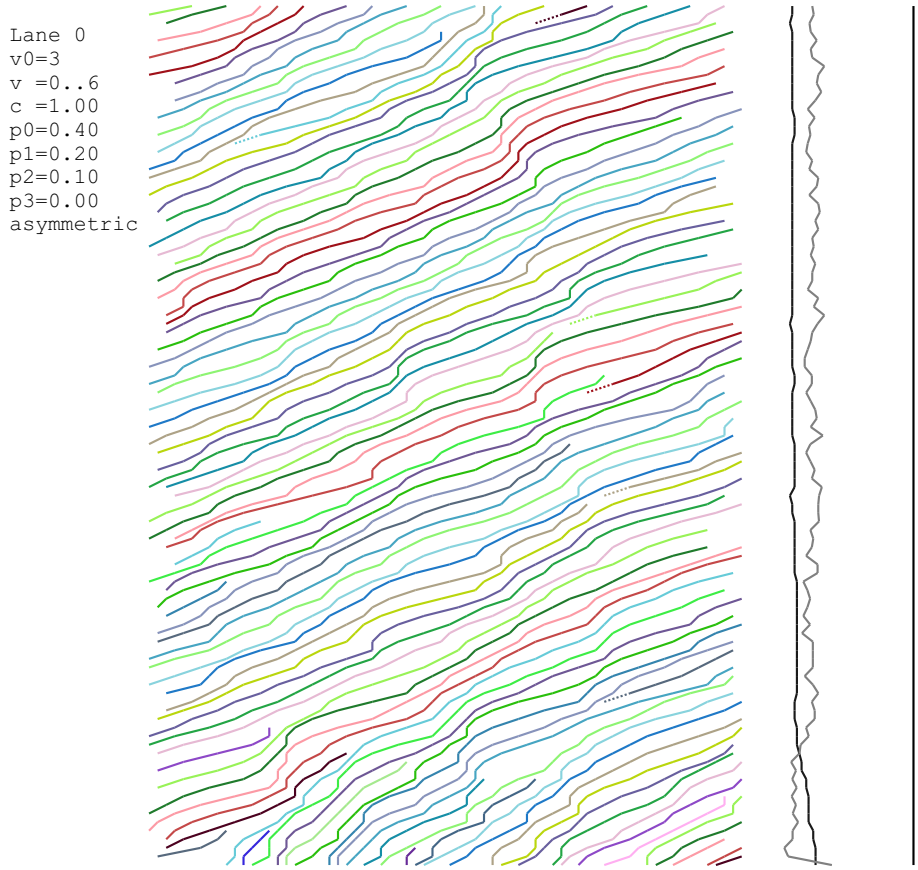

Figure 2.4: An example space-time diagram showing individual trajectories. The flow direction of the vehicles is from left to right in the horizontal direction. The vertical direction pointing upward represents the time. The average speed of the vehicles and the vehicle density in the lane are drawn in gray and black line segments on a linear scale respectively. The maximum speed $v_{\max} = 6$ and the maximum density 1 are represented by the straight line segment on the right side. The end of the vehicle traces refers to the minimum density 0 and the lowest average speed (that is, 0), without a separate line indicator.

diagrams of some particular test cases in a traffic system.

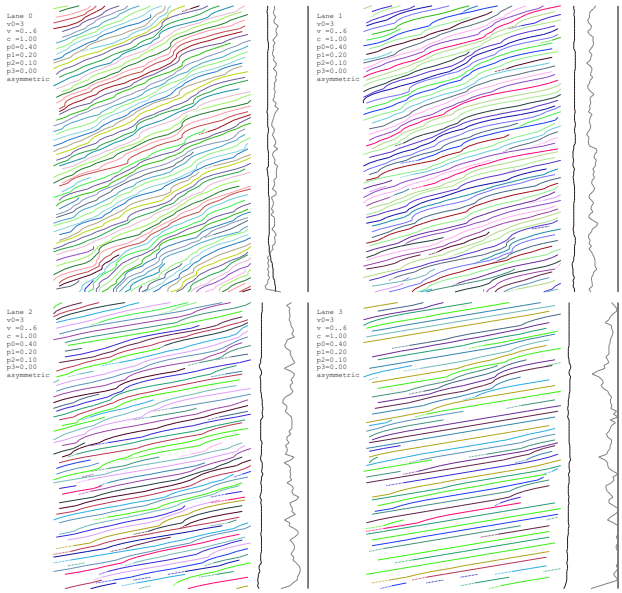

Figure 2.5: Space-time diagram of four asymmetric lanes with $p_0 = 0.4$, $p_1 = 0.2$, $p_2 = 0.1$, $p_4 = 0$ and minimum safety distance. The subfigure associated with L_0 (upper left) is identical with Figure 2.4.

Figure 2.5 shows the case of relatively high initial lane densities in the inner lanes: $p_0 = 0.4$, $p_1 = 0.2$, $p_2 = 0.1$ and $p_3 = 0.0$, with minimum safety distance ($c = 1$, cf. (2.11)). Under the asymmetric circumstances, vehicles in the inner lanes have been gradually shifted into the outer lanes, this can be identified by the slight increase in average speed in L_0 and L_1 (although local congestion may still persist) and the slight decrease in average speed in L_2 at the same time. In L_3, free flow of vehicles can be roughly maintained at a high average speed.

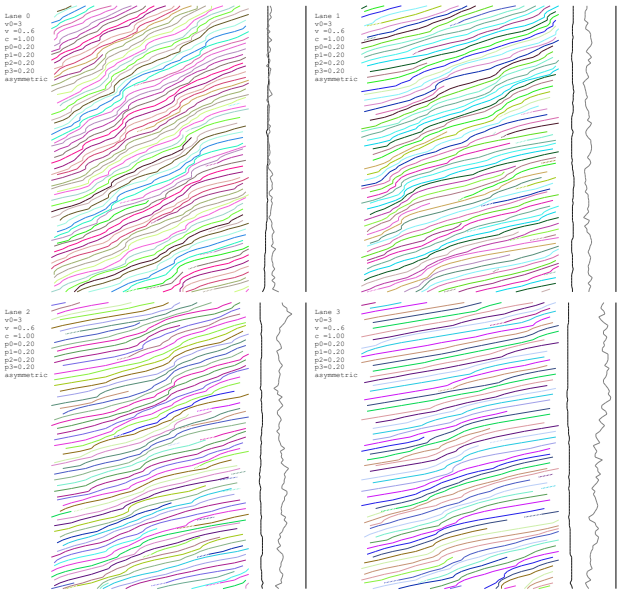

Figure 2.6: Space-time diagram of four asymmetric lanes with $p_0 = p_1 = p_2 = p_3 = 0.2$ and minimum safety distance.

Figure 2.6 shows a test case of equal initial densities $p_0 = p_1 = p_2 = p_3 = p_4 = 0.2$ with minimum safety distance. Under the same asymmetric circumstances, the equal initial lane densities account for the increase in the utilization in inner lanes. In fact, slight decrease in average speed (accompanied by increase in density) in L_0 and L_1 can be observed, whereas L_2 and L_3 show the reverse. In addition, dissolution of the slight congestion in the initial stage can be observed in L_3.

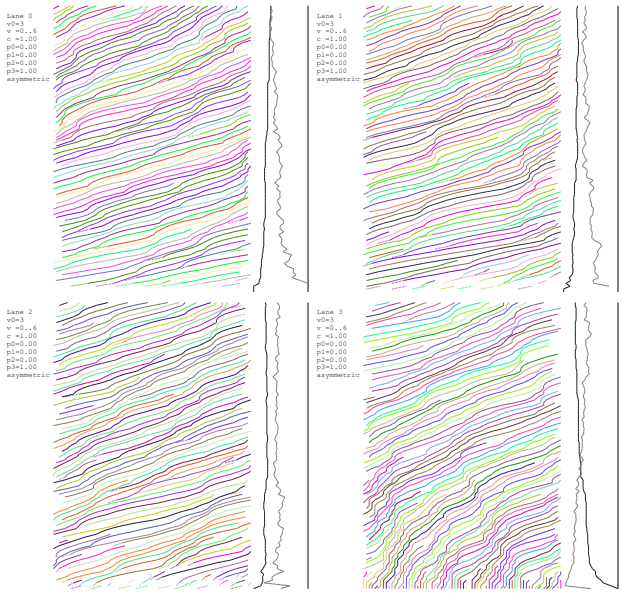

Figure 2.7: Space-time diagram of four asymmetric lanes with $p_0 = p_1 = p_2 = 0$, $p_3 = 1$ and minimum safety distance.

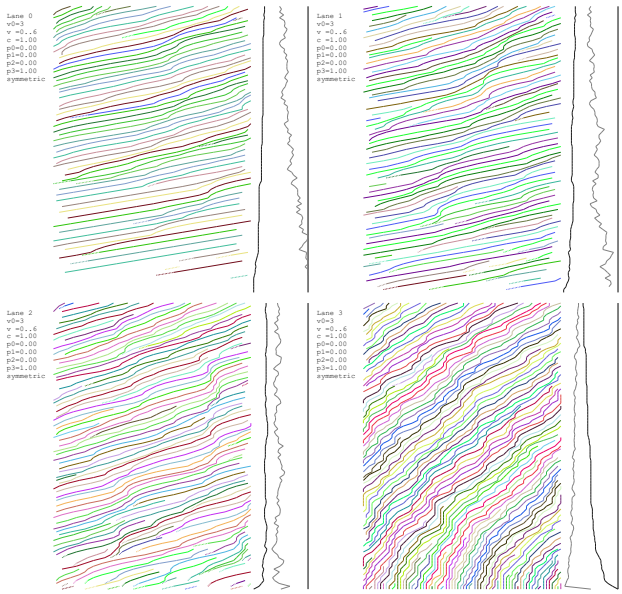

Figure 2.8: Space-time diagram of four symmetric lanes with $p_0 = p_1 = p_2 = 0$, $p_3 = 1$ and minimum safety distance (cf. Figure 2.7).

Figure 2.7 shows an extreme case in the asymmetric case with initial densities $p_0 = p_1 = p_2 = 0$ and $p_4 = 1$ in which the transition of vehicles from outer to inner lanes can be clearly identified: while L_3 is initialized to be fully populated, the last subfigure (lower right) shows that the congestion there becomes gradually dissolved, the heavy demand on road resources is transferred to the other lanes.

The same initial densities have been deployed in a symmetric system, as shown in Figure 2.8. We see that the redistribution of the road resources takes place at a much slower pace compared to the test illustrated in Figure 2.7, since vehicles are not requested to change into inner lanes explicitly.

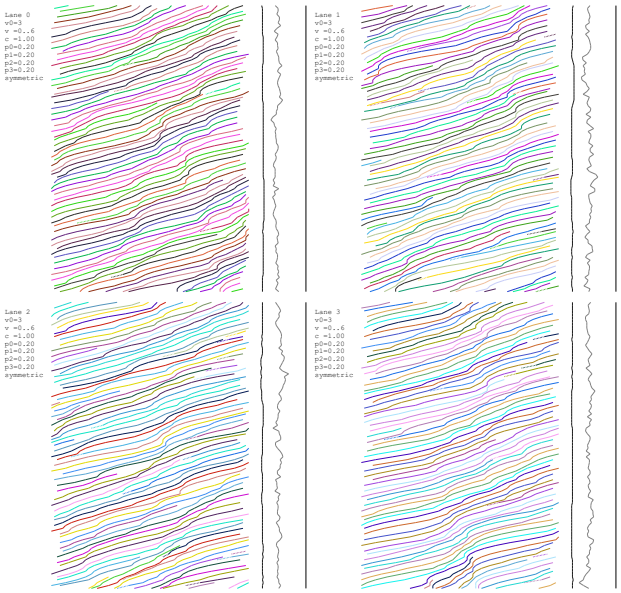

Figure 2.9: Space-time diagram of four symmetric lanes with $p_0 = p_1 = p_2 = p_3 = 0.2$ and minimum safety distance (cf. Figure 2.6).

For $p_0 = p_1 = p_2 = p_3 = 0.2$, Figure 2.9 presents a test case with symmetric conditions. We observe that, compared with the asymmetric case (see Figure 2.6), the average lane speeds now tend to be much closer to each other. Figure 2.10 illustrates an asymmetric test case initialized with the same densities but with doubled safety distance ($c = 2$). Compared with the case of minimum safety distance (see Figure 2.6), the average speeds in the outer lanes become substantially lower, since the safety distance policy is more effective on vehicles with higher speeds.

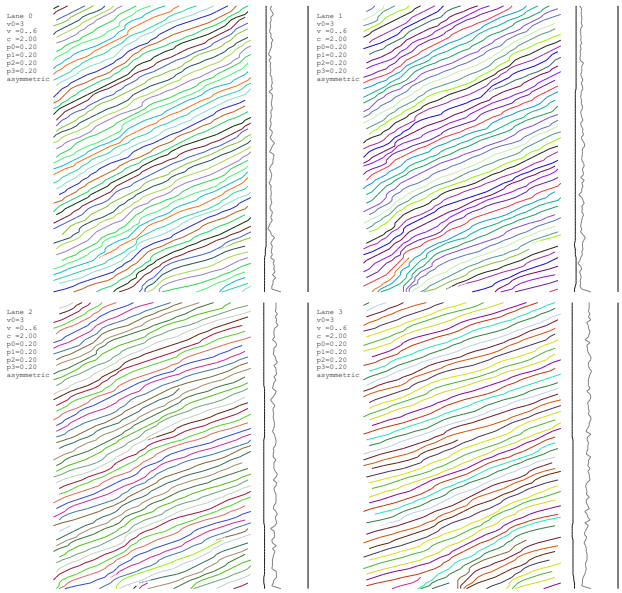

Figure 2.10: Space-time diagram of four asymmetric lanes with $p_0 = p_1 = p_2 = p_3 = 0.2$ and doubled safety distance (cf. Figure 2.6).

2.3.5.2 Asymmetric vs. symmetric driving with various safety distances

Now we study the overall vehicle speed in both asymmetric and symmetric systems more closely. In this section, exactly three lanes will be deployed; we believe that decreasing the number of lanes to a simplest yet nontrivial case helps to demonstrate the difference among these. The initial lane densities p_0, p_1 and p_2 have been selected on an equal basis in the interval $[0, 1]$, excluding $p_0 = p_1 = p_2 = 0$ and $p_0 = p_1 = p_2 = 1$. In the subfigures of Figure 2.11, sample points representing average speed values in both systems have been collected under the condition of various safety distances. On the axes, "`0`" and "`1`" refer to the minimum speed 0 and maximum speed $v_{\max}$ respectively. Average speeds of all three lanes measured in both systems are recorded as the coordinates of the sample points. The colour of the sample points is defined by the initial lane densities p_0, p_1 and p_2.[8]

We see from Figure 2.11 that, generally speaking, the speed difference between asymmetric and symmetric driving is very small. From this, it would be safe to say that there is no substantial difference in overall capacity between asymmetric and symmetric systems. And in reality, both exist.

With larger safety distances $c = 2, 3, 4$ (see (2.11)), we observe that the average speed is slightly higher in the asymmetric case. In the subfigures of Fig-

[8]With p_0, p_1 and p_2 representing the red, green and blue component respectively (red: $(1, 0, 0)$; green: $(0, 1, 0)$; blue: $(0, 0, 1)$; white: $(1, 1, 1)$; black: $(0, 0, 0)$ etc.)

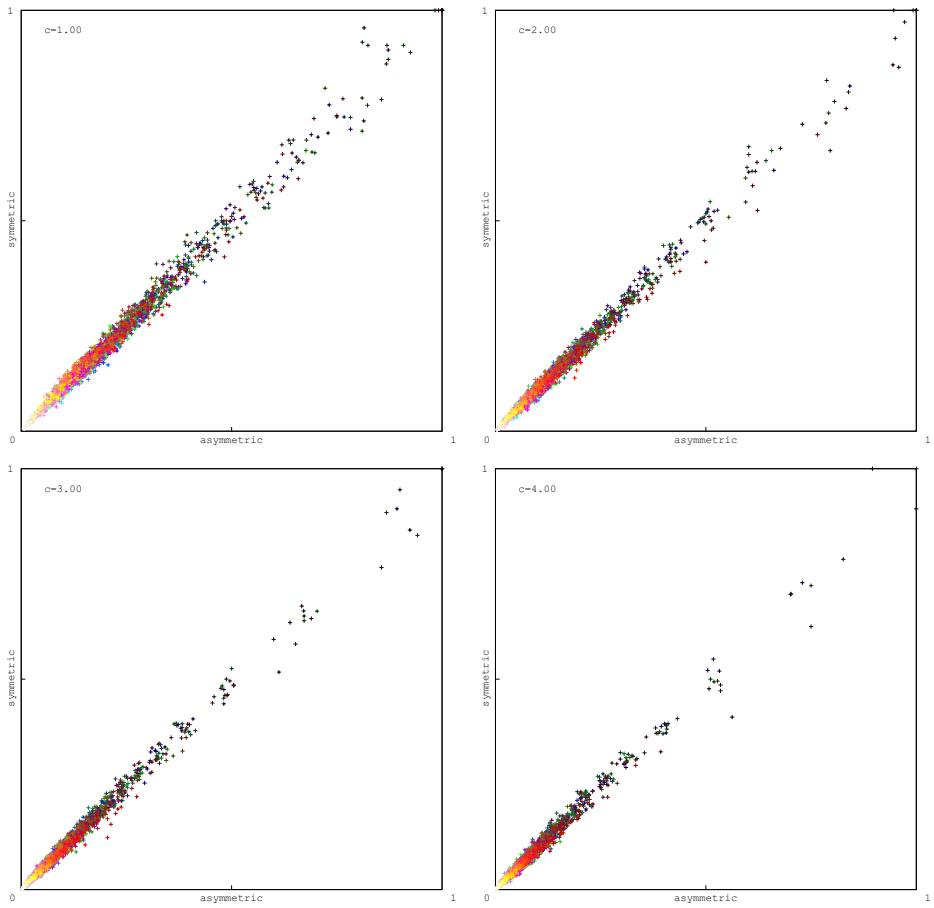

Figure 2.11: Comparison of average speed in asymmetric and symmetric systems, in combination with different safety distances. Both axes refer to average speed measured in a relative linear scale.

ure 2.11 respecting $c = 2, 3, 4$, this can be seen from the sample points diverging from the diagonal. This is also the case with those sample points with a larger red component, these sample points are associated with a higher initial density p_0 in the innermost lane L_0, while p_1 and p_2 are low. Under these circumstances, the vehicles are shifted into outer lanes and the overall road resources can be used effectively. The very light-coloured sample points (seen close to the origin in the subfigures) refer to configurations of very high initial densities: the extreme case $p_0 = p_1 = p_2 = 1$ would lead to complete congestion and the sample point would be of colour white; similarly, $(p_0, p_1, p_2) = (1, 1, 0)$ refers to the case of two fully populated inner lanes L_0 and L_1, the corresponding sample point has the colour yellow[9], in such a case, the overall average speed is very low too.

[9]Initial density configurations associated with colours close to cyan $(p_0, p_1, p_2) = (0, 1, 1)$ or magenta $(p_0, p_1, p_2) = (1, 0, 1)$ are extremely rare in real-world situations, but they are still included in the test.

2.3.5.3 Density, speed and flow

We consider the relationship of density and speed more closely. The product of vehicle density and average speed gives (an estimate of) the number of vehicles passing in a given time period (cf. (1.3)) and is therefore called *flow* of the traffic. The relationship of density and flow has been named *fundamental diagram* of density and flow; it is often shortened as fundamental diagram. In the earliest days, a linear relationship between speed and density had been conjectured; this linearity would imply a parabolic relationship between speed (or density) and traffic flow [38]. Generally speaking, fundamental diagrams contain an upstream sector in which the low density induces a free flow of the traffic and consequently an almost linear increase in the overall traffic flow, and a downstream sector in which the flow decreases owing to the congestion accompanied by the high vehicle density, after once the maximum of the road capacity is reached. [23] states (page 87) that the transition from upstream to downstream sector is rather soft, as in that stage the change in flow volume may not be much correlated with density. It is our opinion that there is no universal fundamental diagram which is valid for all traffic structures.

2.3.5.3.1 Lane-based measurements By now traffic density and vehicle average speed have been measured in the separate lanes. Here we choose to deploy four lanes in the simulation system—mainly to limit the number of the sample point patterns for a clear graphic presentation—and test these with various parameter settings.

In Figure 2.12 we show the relationship of density and speed with minimum safety distance and doubled safety distance in both asymmetric and symmetric cases. The initial lane densities have been selected on a random basis. In the horizontal axis, measured in a relative linear scale, the maximum lane density labelled as "1" refers to approximately 67 vehicles per kilometre (that is $\frac{1}{l_c}$, see page 24). The maximum speed "1" in the vertical axis refers to $135\,\mathrm{km}\cdot\mathrm{h}^{-1}$ (see page 24). We see from the subfigures that there is largely an inverse relationship between average speed and density after an initial stage of free flow in which the maximum speed can be attained under the condition of very low densities. In addition, in the asymmetric case, outer lane sample points (that is, in L_2 and L_3, coloured in green and red) are primarily collected with lower densities; in other words, inner lane sample points are associated with comparatively higher densities, since these will be more efficiently utilized by low-speed vehicles. Sample points in the symmetric case are more evenly distributed in the whole range of lane density.

In Figure 2.13, sample points representing density and flow volume are shown. The maximum flow "1" in the vertical axis refers to the product of maximum density and maximum speed $\frac{v_{\max}}{l_c} = \frac{135\,\mathrm{km}\cdot\mathrm{h}^{-1}}{15\,\mathrm{m}} = 9000$ vehicles per

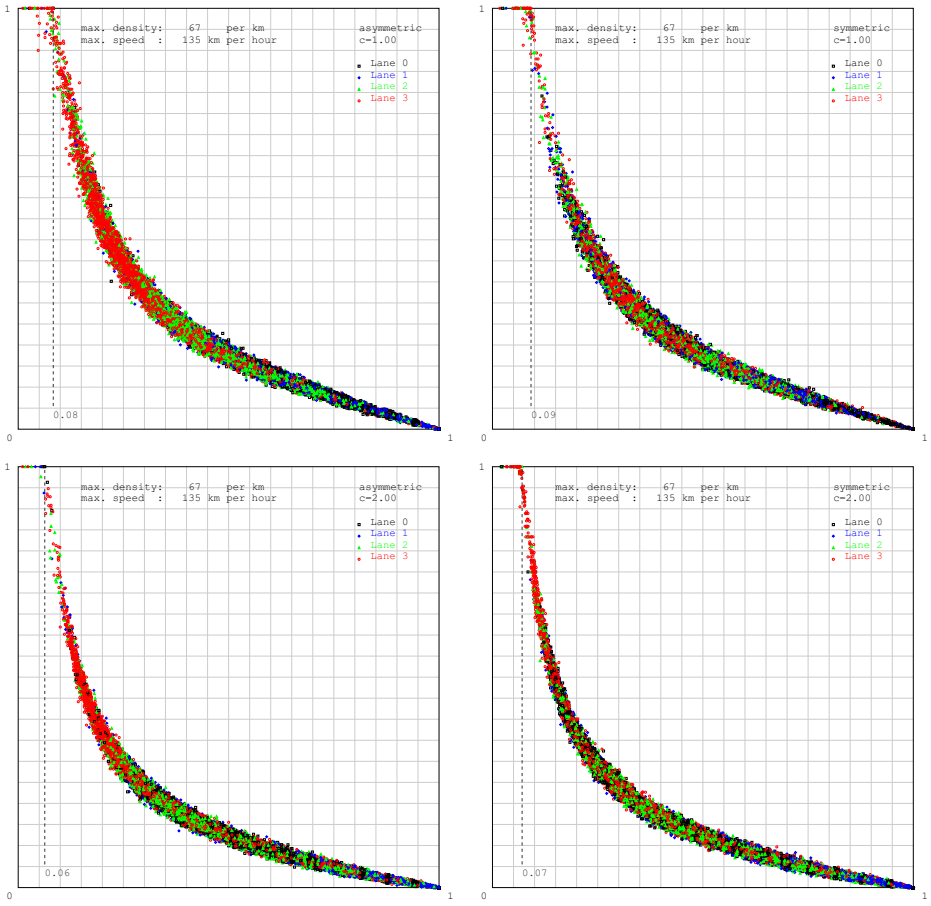

Figure 2.12: Sample points of density and speed, with $c = 1, 2$ in both asymmetric (in the left column) and symmetric (in the right column) cases. The dashed line in the subfigures records the highest local lane density in which the maximum average speed can be achieved. Both axes are on a relative linear scale. In both cases "1" refers to the maximum theoretical value.

hour. This maximum flow volume as a theoretical value is not attainable. In the subfigures, the transition from free flow to reduction of flow and finally to total congestion can be seen. We also observe that in the asymmetric case (in the left column), the free flow situation occurs more frequently in the outer lanes (with sample points coloured in red and green for L_3 and L_2 respectively), which is well confirmed by the nature of the asymmetric traffic. Larger safety distances prevent higher lane densities, therefore, fewer sample points can be collected in the stage of free flow.

Technical note. When the relative scale in this section is applied, the overall shape of the fundamental diagram of density and flow becomes "flatter" with the increase of the safety distance coefficient c. We notice that with constant l_v, l_c, Δt and c, the shape

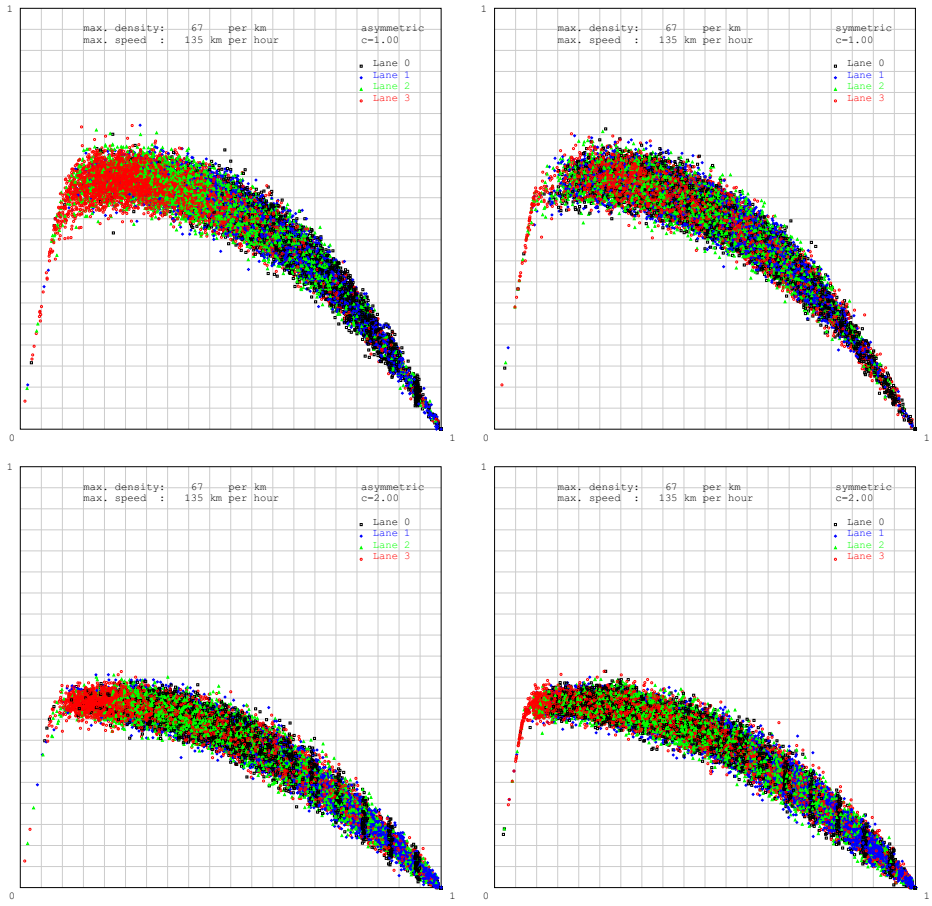

Figure 2.13: Sample points of density and flow, with $c = 1, 2$, in both asymmetric (in the left column) and symmetric (in the right column) cases.

of the overall density-flow diagram is not affected by the lane length or the discrete maximum speed $v_{\max}$. Changing $v_{\max}$ affects the scale in the vertical axis only.

2.3.5.3.2 System-wide measurements and calibration On a system-wide basis, for n lanes with the same length, given ϱ_i and $\bar{v}_i$ as the density and average speed in L_i, $i = 0, \ldots, n-1$, $\bar{\varrho} = \frac{\sum_i \varrho_i}{n}$ and $\bar{v} = \frac{\sum_i \bar{v}_i}{n}$ refer to the average density and speed in the n lanes respectively. The system-wide measurement of $\bar{v}$ is defined to be the average of speed in the lanes, since in a closed system, the overall density remains constant.

In Figure 2.14 we show the results based on system-wide measurements. The results under various safety distance settings are shown as sample points of flow volume in small intervals of density and speed. The maximum flow of over 5000 vehicles per hour is reached at a density of somehow 17 vehicles per kilometre under the condition of minimum safety distance ($c = 1$); the maximum

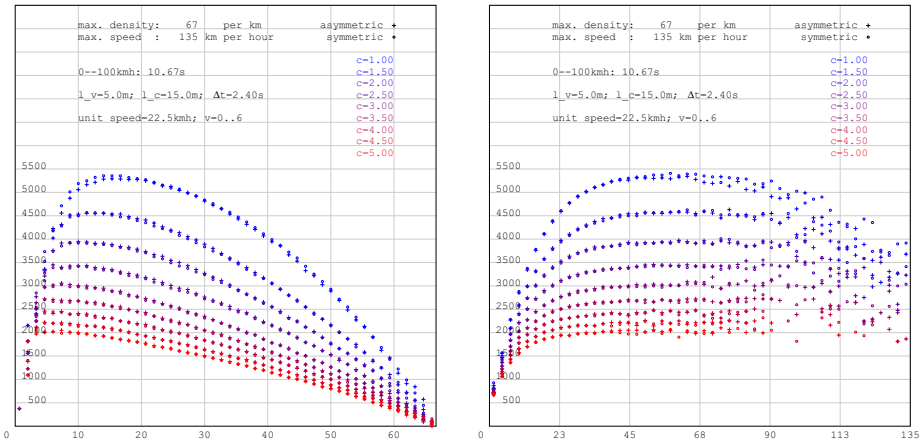

Figure 2.14: Fundamental diagram of standard configuration with various safety distance settings in both asymmetric and symmetric cases. Left: Diagram of density and median flow volume. Right: Diagram of speed and median flow volume. Relative scale is applied with additional labelling of the actual quantities. Vertical axis: Flow volume (number of vehicles per hour). Horizontal axis in the left column: Density (number of vehicles per kilometre). Horizontal axis in the right column: Vehicle speed (in $\mathrm{km} \cdot \mathrm{h}^{-1}$).

flow gradually lowers to about 2000 vehicles per hour at a density of roughly 5 vehicles per kilometre with $c = 5$. In addition, it can be seen in Figure 2.14 that the overall difference between asymmetric and symmetric cases is insignificant. For the sake of comparison with existing empirical data, median flow volume values are presented in the vertical axis.

Figure 2.15 shows results collected from empirical data. It is seen there that an average peak flow of roughly 2300 vehicles per hour is reached at a density of 20 to 30 vehicles per kilometre. Source 2 of Figure 2.15 mentioned that 13% of the recorded vehicles had been trucks (of larger sizes and lower speeds), this would imply a higher density and flow volume, if only vehicles of the size of passenger cars were to be considered.

It is imaginable that with $c = 1$, the flow volume would be much higher than expected. Here two facts are not to be neglected. Firstly, under normal traffic conditions, drivers keep a significantly larger distance to others than the mandatory minimum safety distance; and secondly, the road capacity is not at all time completely exhausted. Combining these two factors, $c \in [2, 4]$ would be a reasonable range for safety distance coefficient. By now, all parameters have been borrowed or deducted from existing sources (see discussion in § 2.3.1). We adjust two system parameters for the calibration of our model. We set $l_v = 4.7\,\mathrm{m}$, this represents the average length of a mid-sized passenger and might therefore reflect the average size of the vehicles more precisely; at the same time, in alignment with the empirical data (see Figure 2.15), maximum lane density is now defined

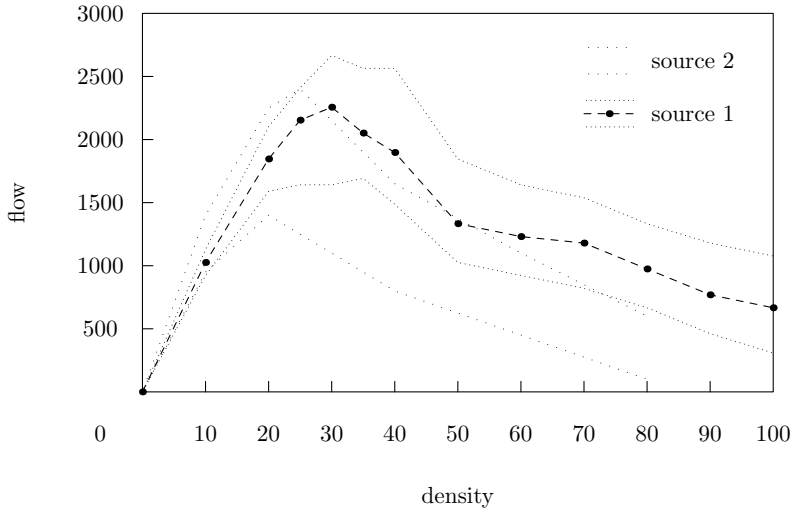

Figure 2.15: A coarse re-plot of two fundamental diagrams collected from empirical data (two different two-lane highways, both with speed limit $120\,\mathrm{km}\cdot\mathrm{h}^{-1}$). Two areas roughly covering the original sample points are shown. Vertical axis: Flow volume (number of vehicles per hour). Horizontal axis: Density (number of vehicles per kilometre). Source 1: [23] (page 87), original sample points are roughly covered in the dotted region, median flow values with various densities are shown in black dots. Source 2: [48], no median flow value had been provided, the original quantities referred to measurements made in two lanes, these have been adjusted in the current plot.

to be 100 vehicles per kilometre, this leads to $l_c = 10\,\mathrm{m}$. The result of this configuration is given in Figure 2.16. Compared to empirical data, it seems that the peak flow is reached too early.

Let u denote the physical speed associated with the discrete unit speed $v = 1$, we have:

$$u = \frac{l_c}{\Delta t}. \tag{2.17}$$

Given a target speed limit v_{target} (for example, $v_{\mathrm{target}} = 120\,\mathrm{km}\cdot\mathrm{h}^{-1}$), the maximum discrete speed is

$$v_{\max} = \left\lceil \frac{v_{\mathrm{target}}}{u} \right\rceil. \tag{2.18}$$

whereas $\lceil\cdot\rceil$ refers to the smallest integer no less than the argument.

Let T denote the time length of acceleration from 0 to $100\,\mathrm{km}\cdot\mathrm{h}^{-1}$ under maximum power. We notice that $\frac{u}{\Delta t}$ refers to the acceleration of physical speed u within a simulation cycle, which in turn is the acceleration of discrete speed 1 per simulation cycle, furthermore, the latter is the maximum possible acceleration in

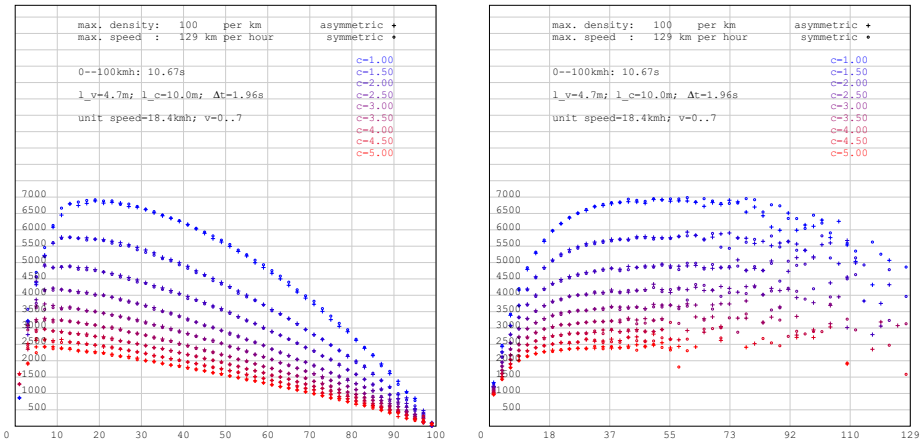

Figure 2.16: Fundamental diagram of median flow volume, density and speed with slightly modified parameters. Owing to the rounding of $v_{\max}$, the real speed limit as product of unit speed u and $v_{\max}$ may be higher than the target speed limit $120\,\mathrm{km}\cdot\mathrm{h}^{-1}$ (see horizontal axis in the right subfigure).

our discrete modelling. Combining these facts, we have:

$$T = \frac{100\,\mathrm{km}\cdot\mathrm{h}^{-1}}{\frac{u}{\Delta t}}.$$

By (2.17), we have

$$(\Delta t)^2 = \frac{l_c \cdot T}{100\,\mathrm{km}\cdot\mathrm{h}^{-1}},$$

this is

$$\Delta t = \sqrt{\frac{3.6\,\mathrm{s}\cdot l_c \cdot T}{100\,\mathrm{m}}}. \tag{2.19}$$

Given l_c measured in metres and T in seconds, Δt in seconds can be easily calculated.

It is to be mentioned that increasing T leads to a higher flow volume, although a larger T implies a weaker acceleration capacity of the vehicles. The explanation is that in the flow diagram with axes of relative scale, the actual flow volume respecting a specific configuration (of l_v, l_c and Δt) is related with the shape (flatness) of the density-flow curve represented by the sample points. With a constant configuration (of l_v, l_c and Δt), varying $v_{\max}$ has no effect on the shape of the density-flow curve, nevertheless the flow volume changes accordingly, since the maximum flow on the vertical axis is the product of maximum density and maximum speed. On the other hand, by (2.19), Δt is correlated with the square root of T; by (2.17) and (2.18) we know that $v_{\max}$ is correlated with Δt. Therefore, the actual flow volume increases with the enlargement of T, cf. Figure 2.17.

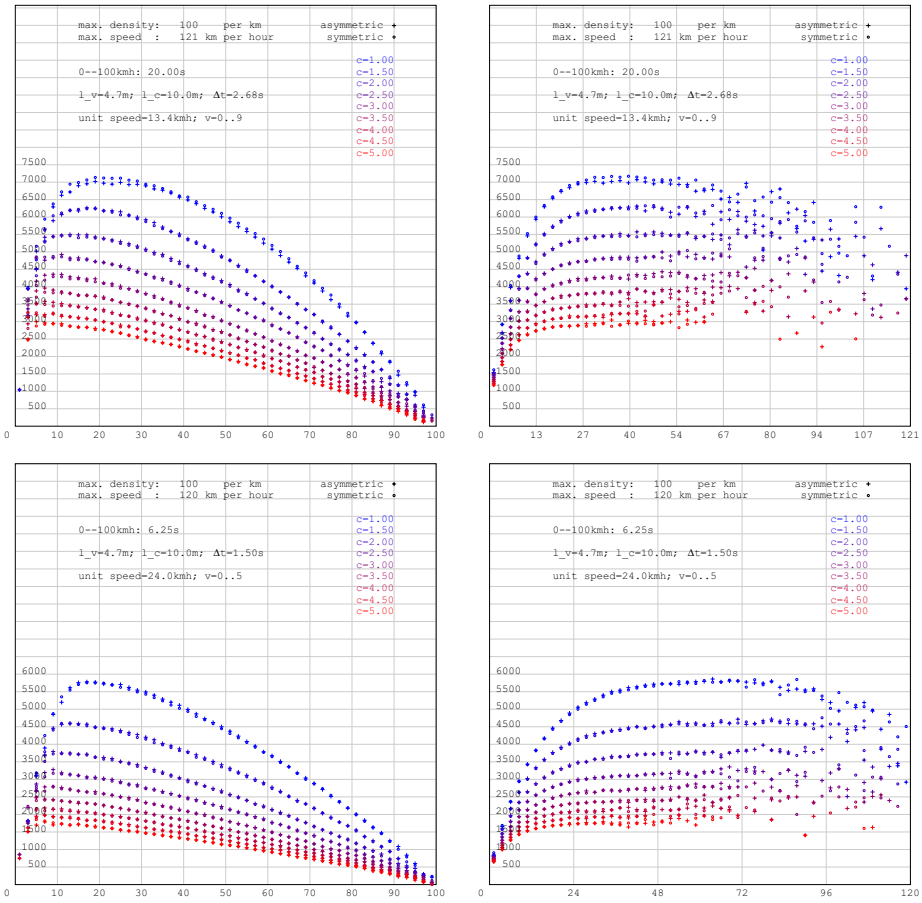

Figure 2.17: Fundamental diagrams of median flow volume, density and speed with different values of T. First row: T is larger than its counterpart in the standard configuration. Second row: T is reduced to produce the correct flow volume.

The flow volume produced by our initial configuration of $\Delta t = 2.4\,\mathrm{s}$ is too high. From the above discussion we know that a smaller Δt would deliver better simulation results. Indeed, with $\Delta t = 1.5\,\mathrm{s}$ (l_v has already been modified from $5\,\mathrm{m}$ to $4.7\,\mathrm{m}$ and l_c is $10\,\mathrm{m}$ instead of $15\,\mathrm{m}$) flow volume in a range close to that of the empirical data can be produced. However, $\Delta t = 1.5\,\mathrm{s}$ implies $T = 6.25\,\mathrm{s}$ which represents a much too high acceleration capacity, cf. the second test case in Figure 2.17. As a solution for this, local acceleration (cf. Figures 2.2 and 2.3) needs to be adjusted. For this purpose, we introduce an additional acceleration multiplier s ($s > 1$) that the vehicle should have a simulated acceleration capacity from 0 to $100\,\mathrm{km} \cdot \mathrm{h}^{-1}$ in a time length of Ts. To achieve this, we request that (2.4a) will be carried out with a probability of q_v, for $v = 0, \ldots, v_{\max} - 1$. Expressed in discrete form, the acceleration from $v = 0$ to $v = v_{\max}$ should then

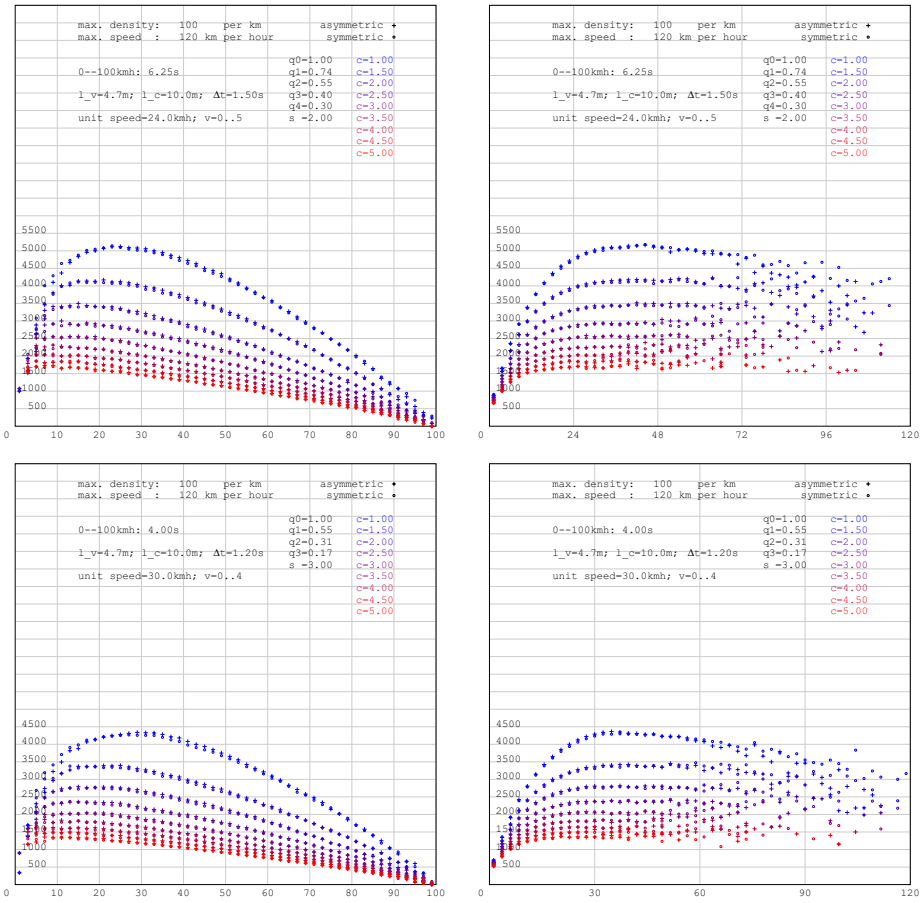

Figure 2.18: Calibrated diagrams of median flow volume, density and speed with additional acceleration multipliers.

take $v_{\max} \cdot s$ steps, that is

$$\sum_{i=0}^{v_{\max}-1} \frac{1}{q_i} = v_{\max} \cdot s. \tag{2.20}$$

In the trivial case $v_{\max} = 1$ (in which the vehicle speed will not be differentiated in the simulation) we may set: $q_0 = \frac{1}{s}$. Otherwise, we request

$$q_0 = 1, \tag{2.21}$$

$$q_{i+1} = q_i \cdot q, \quad \text{for } i = 0, \ldots, v_{\max} - 2. \tag{2.22}$$

(2.21) says that acceleration from a stationary state will always (except for the above-mentioned trivial case) be carried out; and with (2.22), acceleration decays

with a probability q ($0 < q < 1$). Rewriting (2.20), we have

$$1 + q^{-1} + \cdots + q^{1-v_{\max}} = \frac{q^{-v_{\max}} - 1}{q^{-1} - 1} = v_{\max} \cdot s. \tag{2.23}$$

With the solution of (2.23) the prolonged acceleration can be simulated. Two further test cases are shown in Figure 2.18. For the configuration of $T = 6.25\,\mathrm{s}$ and $s = 2$, $c = 3$ or 3.5 seems to produce a maximum flow volume in the range of 2300 to 2600 vehicles per hour; for $T = 4\,\mathrm{s}$ (in this case, Δt is further reduced to be $1.2\,\mathrm{s}$) and $s = 3$, $c = 2.5$ renders a maximum flow volume of approximately 2300 per hour at a density slightly above 20 vehicles per kilometre. These two parameter settings of T and s produce a acceleration capacity (0 to $100\,\mathrm{km \cdot h^{-1}}$ in 12.5 and 12 seconds respectively) close to reality and at the same time, $v_{\max}$ in (2.18) has integer solution.

Remark: In the classical model [47] and its extensions, the braking behaviour of the (drivers of the) vehicles is regulated by an empirical probability number (cf. (2.5)). In contrast, in our model, deceleration is controlled by a series of probability numbers derived from physical quantities. In addition, it is to be mentioned that deceleration (or braking-down) is not to be understood literally; in our modelling it simply refers to the physical state of a vehicle under its maximum technical performance which can be the result of the traffic conditions or the subjective decision of its driver.

Here we proposed a discrete model for vehicular traffic with generically multiple lanes with modifiable safety distances. The model has been calibrated to produce results close to empirical data. The background geometry of the simulation system has been defined on a regular grid. We will encounter similar geometries in the next chapter for the simulation of pedestrian dynamics.

Chapter 3

Grid-based methods for pedestrian dynamics

Similar to vehicular traffic, pedestrian dynamics can be modelled on discrete space as well. Pedestrian behaviour can be described on three different levels (see § 1.3.1). On the tactical level, we need to know how the position transition (see § 1.3.3) can be planned in a system of simulation objects which interact with each other; and on the operational level, how this can be realized on the underlying geometry.

3.1 Square Grid

How to describe the position transition $(\Delta x, \Delta y)$ in general is not so simple as it might appear to be. The variables x, y in $\mathbb{R}^2$ mentioned in (1.19) are frequently approximated on a homogeneous square grid in $\mathbb{R}^2$. We understand a homogeneous square grid Ω as a set of index pairs $\{0, \ldots, n_x - 1\} \times \{0, \ldots, n_y - 1\}$ with $n_x, n_y \in \mathbb{N}^+$ denoting the number of indices in x- and y-direction respectively. Given any $(x_0, y_0) \in \mathbb{R}^2$ and $d \in \mathbb{R}^+$, an index pair (x, y) defines a two-dimensional interval

$$\big[x_0 + x \cdot d, x_0 + (x+1) \cdot d\big) \times \big[y_0 + y \cdot d, y_0 + (y+1) \cdot d\big), \tag{3.1}$$

for $x, y \in \mathbb{N}$. The interval of (3.1) is left open on the right side to ensure that each position (x, y) can only be in the range of one such interval.

The grid Ω thus refers to the two-dimensional region of $\big[x_0, x_0 + n_x \cdot d\big) \times \big[y_0, y_0 + n_y \cdot d\big)$. For simplicity's sake, sometimes the interval referred to by an index pair (x, y) will be shorthanded as interval (x, y). In the context of discrete modelling, the interval (3.1) is often revised to be closed on the right side as well; these fully closed two-dimensional intervals will be called *grid cell*s or simply *cell*s.[1] For $x = 0, \ldots, n_x - 1$ and $y = 0, \ldots, n_y - 1$, the coordinates $(x_0 + x \cdot d, y_0 + y \cdot d)$ will be called *grid node*s or simply *node*s. Two grid cells may have one or more grid nodes in common, in this case, the two grid cells are

[1] In comparison, in the context of cellular automaton (see § 2.1.1), cells are requested to be homogeneous, no specific geometric definition is needed.

said to be *neighbour*s. If two grid cells have exactly two nodes in common, the straight line segment connecting these two points forms a common *edge* shared by the two grid cells, in this case, these two grid cells are *immediate* neighbours with each other.

3.1.1 Perspective from computer graphics

The approximation of position transition on a square grid can be considered from the perspective of computer graphics. The simplest representation of a position transition is the straight line segment connecting the two points by which this transition is defined. In discrete form, Δx and Δy are assumed to be integers.

3.1.1.1 A simple method

Let $\Delta x > 0$. We consider a very simple method of drawing a line segment from position (x_1, y_1) to a new position (x_2, y_2). This method relies on the linear interpolation of y with $x = x_1 + 1, \ldots, x_2 - 1$ as supporting points. The coordinates of the supporting points on the square grid are composed of the x-values and the corresponding interpolated values of y, the latter interpolation values are rounded, if necessary.

```
/* a simple method for straight line drawing     */
/* from (x1, y1) to (x2, y2) by interpolation in */
/* the x-direction, intermediate results will be */
/* passed to output_index_pair(int, int)         */
void simple_line_by_x(int x1, int y1,
                      int x2, int y2) {

    assert(x2 > x1);

    double k = ((double)(y2 - y1)) / ((double)(x2 - x1));
    double y =  (double) y1;

    for (int x = x1+1; x <= x2; x++) {

        /* interpolation, equal to */
        /* y = y1 + k*(x - x1);    */
        y += k;

        output_index_pair(x, round(y));
    }
}
```

The above pseudocode does not include the index pair of the start position (x_1, y_1) in the output, this has the effect that in a sequence of position transi-

tions, no position will be counted twice. See Figure 3.1 for a few simple examples.

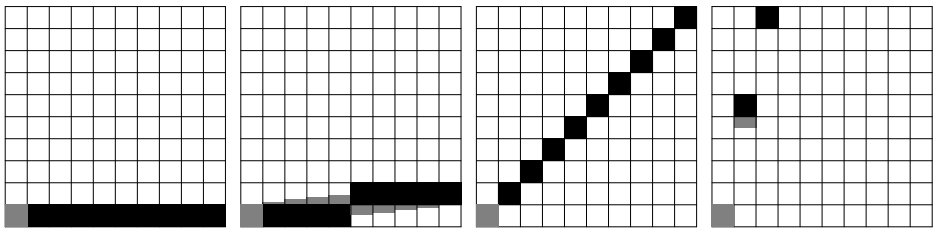

Figure 3.1: Examples of the simple method: $(\Delta x, \Delta y) = (9,0), (9,1), (9,9)$ and $(2,9)$. The start position $(0,0)$ is drawn in gray. The grid cells translated by the interpolated y-values without rounding are also drawn in gray.

It is clear that this method works awkwardly when Δy has a much larger magnitude than Δx. To bypass this problem, we may swap the axial directions. This means that the interpolation should be carried out in the direction in which more supporting points are available. We refer to the axial direction with a larger position transition as the *main axial direction*.

Assume now x is the main axial direction. A closer look at the above pseudocode reveals that in the loop of x, y either remains unchanged or is incremented by 1. We refer to position transitions $(\Delta x, \Delta y)$ with $\Delta x, \Delta y \in \{-1, 0, 1\}$ and $(\Delta x, \Delta y) \neq (0,0)$ as *elementary steps*. They will be denoted by $M_{+,+}$, $M_{+,0}$, $M_{+,-}$, $M_{0,+}$, $M_{0,-}$, $M_{-,+}$, $M_{-,0}$, $M_{-,-}$, respecting the cases of $\Delta x = 1, \Delta y = 1$ and so on. $M_{0,0}$, for the case of $\Delta x = \Delta y = 0$, will be addressed as *null step*. The simple method in the x-direction allows $M_{+,0}$ and $M_{+,+}$ as valid elementary steps in the loop.

3.1.1.2 Bresenham's algorithm

An improvement to the above-mentioned simple method would be the approximation of the position change $(\Delta x, \Delta y)$ by the grid cell positions that are closest to the straight line segment defined by $(\Delta x, \Delta y)$. This can be traced back to the renowned Bresenham's algorithm for straight line [7].

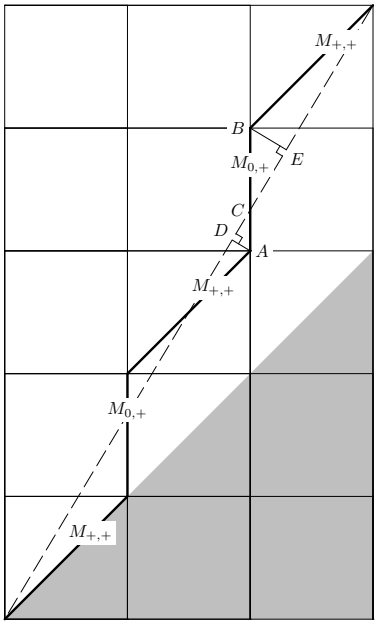

Figure 3.2: Bresenham's algorithm in the second octant with a series of elementary steps for $(\Delta x, \Delta y) = (3, 5)$. $|AD|$ and $|BE|$ are the distances from A and B, respectively, to the line segment to be approximated.

Without loss of generality, let $\Delta y > \Delta x > 0$. The problem is now confined into the second octant. Further in the current section, we let the position (x, y) refer to Cartesian coordinates—instead of position of a grid cell—for a clearer graphic representation. We also notice that, if the position transition $(\Delta x, \Delta y)$ is confined to $\Delta x, \Delta y \in \{-1, 0, 1\}$—excluding $(\Delta x, \Delta y) = (0, 0)$ naturally—the elementary step sequence would degenerate into a single elementary step M, which is exactly the case of "one cell per step" in the common CA models (see later in § 3.1.2). Here we consider the nontrivial case of $(\Delta x, \Delta y)$.

The idea of Bresenham's algorithm is to select the grid cell positions which are the "closest" to the original line segment. Figure 3.2 shows an example of a series of elementary steps. In this example, the first octant, as an open interval, is filled in gray colour, this is to remind that the elementary steps constructed later by (3.2)–(3.4) will always be outside of it.

Let the start position be O. Let A and B be the new position after the elementary step $M_{0,+}$ and $M_{+,+}$ respectively, as Figure 3.3 shows. We will see that the straight line $y = 2x$ plays a important role in deciding which of the elementary steps should be chosen.

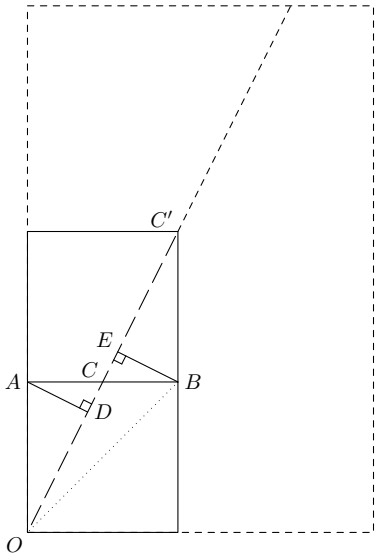

Figure 3.3: A special case of Bresenham's algorithm in the second octant ($\Delta y = 2\Delta x$).

In Figure 3.3, the original line segment is shown as the dashed line segment OC' with its extension. Obviously, the distance from A and B to this line segment is the length of the altitude AD and BE respectively. If the slope of OC' is 2 ($\Delta y = 2\Delta x$), it follows easily on the grid that $|AD| = |BE|$.[2] By comparing $\frac{\Delta y}{\Delta x}$ with 2, we know which elementary step should be carried out.

For this purpose, the algorithm of Bresenham defines the following relation:

$$\nabla_1 = 2\Delta x - \Delta y; \tag{3.2}$$

for $i = 2, \ldots, \Delta y$,

$$\nabla_i = \begin{cases} \nabla_{i-1} + 2\Delta x, & \text{if } \nabla_{i-1} < 0, \\ \nabla_{i-1} + 2\Delta x - 2\Delta y, & \text{if } \nabla_{i-1} \geq 0. \end{cases} \tag{3.3}$$

Depending on ∇, the elementary step M will be given as:

$$M = \begin{cases} M_{0,+}, & \text{if } \nabla_i < 0, \\ M_{+,+}, & \text{if } \nabla_i \geq 0, \end{cases} \tag{3.4}$$

for $i = 1, \ldots, \Delta y$.[3] Bresenham's algorithm runs in the main axial direction of the position transition $(\Delta x, \Delta y)$, that is, in the direction in which more supporting points are available.

Similar to the simple method mentioned in 3.1.1.1, Bresenham's algorithm gives an approximation of the position transition $(\Delta x, \Delta y)$ on the square grid. In fact, the interpolated value associated with each supporting point of the line segment $(\Delta x, \Delta y)$ will be approximated (that is, rounded) on the grid. However, Bresenham's algorithm differs substantially from the former in that no floating point calculation is involved.

For two reasons Bresenham's algorithm is not suitable for the calculation of position transition in pedestrian dynamics. The first is the rigid nature of the sequence of elementary steps associated with $(\Delta x, \Delta y)$. With a given pair

[2] The original algorithm implies that in such case B is closer to the line segment than A and consequently $M_{+,+}$ will be executed as the next elementary step. With consideration of the fact that $\angle BOE < \angle AOE$, this becomes very reasonable.

[3] In other octants, slight modifications will be expected.

$(\Delta x, \Delta y)$, the occurrences of the elementary steps involved are fixed. This often leads to the so-called "deadlock" problem that simulation objects block the passage of one another in a unreasonable manner, even when they are not densely populated in the system at all.

The second disadvantage is subtler, namely the elementary steps are not of an equal geometric length on the grid Ω. Indeed, four of them, which are axis-parallel in the underlying coordinate system, have a length of 1, while the other four, diagonal on the grid, have a length of $\sqrt{2}$. As a result of this, the geometric distance associated with a position transition $(\Delta x, \Delta y)$ may differ from one another greatly. In other words, for position changes of an equal geometric length on the grid, the numbers of the elementary steps required may differ too much. Since for each pedestrian as a simulation object, the accumulated number of the undertaken elementary steps is furthermore associated with the system clock in the simulation, this feature incurs a grave problem.

3.1.2 Transition matrix

Before we continue our discussion on position transition $(\Delta x, \Delta y)$, we go back to a previous topic. In addition to the approaches mentioned in § 2.2, cellular automata have been intensively studied for the simulation of pedestrian dynamics. The CA model [9, 37, 60] has been widely acclaimed for its ability of reproducing various collective pedestrian behavioural phenomena. This model established a basis for many other contributions in this field.

By definition of CA, in every grid cell, at all time, no more than one simulation object (that is, pedestrian) can be present. The individual pedestrians in the simulation will be called *particle*s as in many other microscopic models (cf. § 1.3.2). The grid cell size represents implicitly the "personal space" for the particles. In the modelling of pedestrian dynamics, pedestrians—as particles in the simulation—reserve this minimum space for themselves exclusively; unlike other kinds of molecules, pedestrian particles cannot be further compressed. This fact forms the fundamental basis of all grid-based methods in the modelling of pedestrian dynamics.

The model of [9, 37, 60] applies a fixed grid cell size of $0.4\,\mathrm{m} \cdot 0.4\,\mathrm{m}$ with a parallel update scheme carried out at an time interval of $0.3\,\mathrm{s}$. Thus in this ($v_{\max} = 1$)-model, the pedestrian walking speed is approximately defined as $1.33\,\mathrm{m} \cdot \mathrm{s}^{-1}$, which has been generally accepted as a good estimate of the average pedestrian walking speed for a long time (see [72], page vi). Diagonal moves are allowed for the particles.

Each particle is given a *preference matrix*, a $3 \cdot 3$ matrix showing the particle's tendency of carrying out the corresponding elementary step choices $M_{+,+}, \ldots, M_{-,-}$. In addition, this model applies the notion of *floor field* (see

additional reference [52]) for the calculation of position transitions of the particles. Floor fields can be static or dynamic. The static floor field is independent of time and describes the environment settings in the simulation; the dynamic floor field, however, aims to show the interacting impact made by the particles in the long range. The floor fields can be seen as a modelling component on the tactical level.

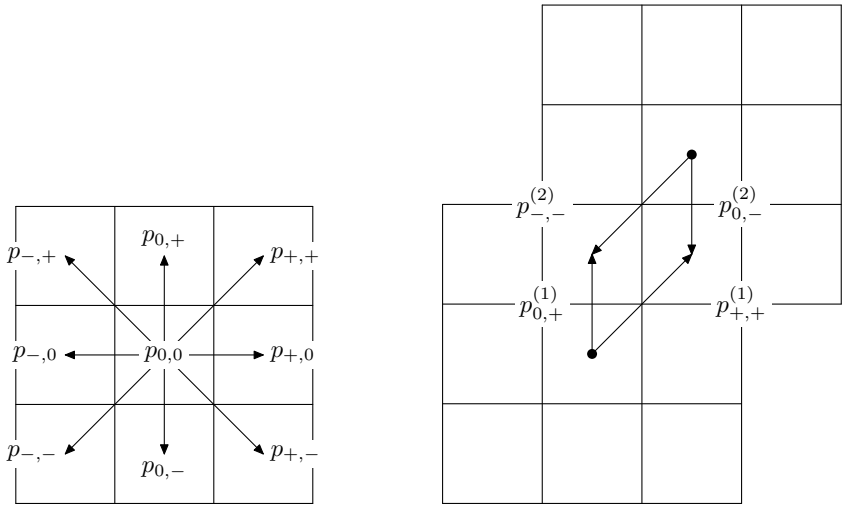

Figure 3.4: The concept of transition probability [9]. Left: Transition probabilities for a single particle. Right: Relating transition probabilities for two particles (addressed by superscripts $^{(1)}$ and $^{(2)}$) in conflict.

The particle's actual step choice is decided by the so-called *transition probability*. A particle's transition probability for a certain elementary step is defined as the product of the related item in this particle's preference matrix with the corresponding matrix items in the floor fields. In this sense, the original preference matrix becomes now the *transition matrix*:

$$P = \begin{pmatrix} p_{-,+} & p_{0,+} & p_{+,+} \\ p_{-,0} & p_{0,0} & p_{+,0} \\ p_{-,-} & p_{0,-} & p_{+,-} \end{pmatrix}. \tag{3.5}$$

$p_{0,0}$ is sometimes re-defined to be 0 (to prevent the null step $M_{0,0}$). As transition probabilities, the items of P in (3.5) are further to be normalized, so that they sum up to 1.

In each simulation cycle, the following rules apply:

1. A target position is selected according to the transition probabilities. If the target is inaccessible, the particle does not move.

2. If the target is accessible and no other particle has the same target, the particle makes the move.

3. If two or more particles have the same target, the particle to make the final move is to be decided by the weights of their transition probabilities. No other particle is allowed to make the move toward the same target.

Rule 3 can be further detailed by the right subfigure of Figure 3.4. Let particles 1 and 2 have nonzero transition probabilities $p^{(1)}_{0,+}$, $p^{(1)}_{+,+}$, $p^{(2)}_{-,-}$ and $p^{(2)}_{0,-}$ respectively. If, by rule 1, particle 1 chooses the elementary step $M_{0,+}$, and at the same time particle 2 attempts a step choice of $M_{-,-}$, there will be a conflict between the two to solve. The probability to grant particle 1 a successful move will be given as $\frac{p^{(1)}_{0,+}}{p^{(1)}_{0,+}+p^{(2)}_{-,-}}$. In analogy, if particle 1 chooses the elementary step $M_{+,+}$, while particle 2 plans to make a move of $M_{0,-}$, the probability for a conflict solution in favour of particle 2 will be given as $\frac{p^{(2)}_{0,-}}{p^{(1)}_{+,+}+p^{(2)}_{0,-}}$ in the current simulation cycle.

After the move of the particle, the dynamic floor field will be renewed. However, this is to happen after all the particles have updated their positions (including those of null steps) in the current simulation cycle. To be more specific, the old position of a particle will leave a "trace" in the dynamic floor field, leading to a value increment of the corresponding matrix items, this trace has the effect of "diffusion" and "decay". Within a pre-defined neighbourhood, the effect of diffusion increases the values of the corresponding floor field matrix items with pre-defined weights, this diffusing trace will affect the particles in the next simulation cycles; the effect of decay is time-dependent and leads to a decrement in the items of the floor field matrix, thus the influence on the other particles in the simulation system diminishes gradually.

Remark: Under the above rules, a particle with more nonzero items in the transition matrix has a smaller chance in a conflict with others to reach the temporal target. In our opinion, once a particle has a target—which is to be understood as the configuration in the current simulation cycle—in conflict with others, the chance of a favouring solution should not be affected by this particle's other possible preferences.

We take a closer look of the situation. For simplicity's sake, we let:

$$p^{(1)}_{0,+} = p_1, \quad p^{(1)}_{+,+} = 1 - p_1, \quad p_1 \in (0,1), \tag{3.6}$$

$$p^{(2)}_{0,-} = p_2, \quad p^{(2)}_{-,-} = 1 - p_2, \quad p_2 \in (0,1). \tag{3.7}$$

This is to say, both particles have exactly two possible steps in the current simulation cycle, some combined step choices result in conflict. All possible situations respecting the two particles are listed in Figure 3.5:

1. First row left: Both particles undertake a vertical move. This case is free

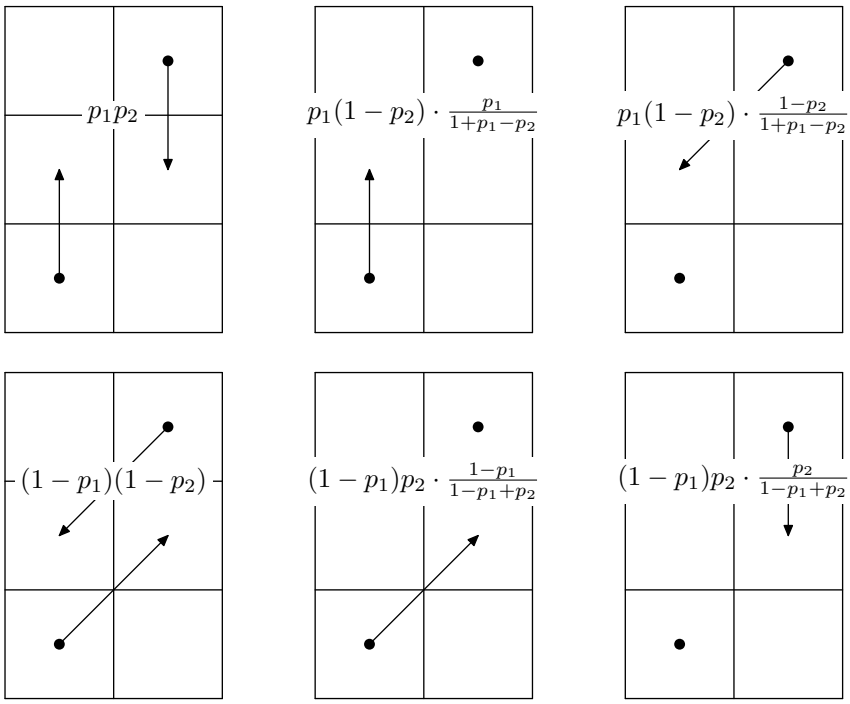

Figure 3.5: Step execution of two particles via transition matrix. First row: Particle 1 targets a vertical move. Second row: Particle 1 targets a diagonal move. Left column: No conflict between the two particles. Middle column: Conflict solution in favour of particle 1. Right column: Conflict solution in favour of particle 2.

of conflict and has a probability for $p_1 p_2$.

2. Second row left: Both particles undertake a diagonal move. This case is free of conflict and has a probability for $(1-p_1)(1-p_2)$.

3a. First row middle: Particle 1 attempts a vertical move and particle 2 a diagonal move, conflict is solved in favour of particle 1. This has a probability for $\frac{p_1^2(1-p_2)}{1+p_1-p_2}$.

3b. First row right: Same as 3a, but solution in favour of particle 2. This has a probability for $\frac{p_1(1-p_2)^2}{1+p_1-p_2}$.

4a. Second row middle: Particle 1 attempts a diagonal move and particle 2 a vertical move, conflict is solved in favour of particle 1. This has a probability for $\frac{(1-p_1)^2 p_2}{1-p_1+p_2}$.

4b. Second row right: Same as 4a, but solution in favour of particle 2. This has a probability for $\frac{(1-p_1)p_2^2}{1-p_1+p_2}$.

Technical note. The extreme case $p_1 = 0, p_2 = 1$ would result in a probability of 50% for both particles in cases 4a and 4b; this would also lead to a division-by-zero error in the evaluation of cases 3a and 3b. However, rule 1 (page 47) excludes these two cases implicitly.

The mathematical expectation of the number of the elementary steps carried out by particle 1 (and particle 2 as well), under the circumstance of a possible diagonal conflict, would be:

$$p^{(1)} = p_1 p_2 + (1-p_1)(1-p_2) + \frac{p_1^2(1-p_2)}{1+p_1-p_2} + \frac{(1-p_1)^2 p_2}{1-p_1+p_2}.$$

The value of $p^{(1)}$ in $(0,1) \times (0,1)$ is shown in Figure 3.6. The mathematical expectation of $p^{(1)}$ on the interval $(0,1) \times (0,1)$, that is, the mathematical expectation of $p^{(1)}$ with consideration of all combinations of possible p_1 and p_2, is[4]:

$$\int_0^1 \int_0^1 \left(xy + (1-x)(1-y) + \frac{x^2(1-y)}{1+x-y} + \frac{(1-x)^2 y}{1-x+y} \right) \mathrm{d}x\, \mathrm{d}y = \frac{3}{4}.$$

[4]To evaluate this mathematical expectation, for the third component of the integral we have:

$$\begin{aligned} \int_0^1 \int_0^1 \frac{x^2(1-y)}{1+x-y} &= \int_0^1 \int_0^1 \left(x^2 - \frac{x^3}{1+x-y} \right) \mathrm{d}x\, \mathrm{d}y \\ &= \int_0^1 \left(x^2 y \Big|_{y=0}^{y=1} + x^3 \log(1+x-y) \Big|_{y=0}^{y=1} \right) \mathrm{d}x \\ &= \int_0^1 \left(x^2 + x^3 \log \frac{x}{1+x} \right) \mathrm{d}x, \end{aligned} \tag{$*$}$$

whereas

$$\int x^3 \log \frac{x}{1+x}\, \mathrm{d}x = \frac{x^3}{4} + \frac{x^2}{8} - \frac{x}{4} + \frac{x^4 \log \frac{x}{x+1}}{4} + \frac{\log(x+1)}{4} + C. \tag{$**$}$$

Noticing

$$\lim_{x \to 0} x \log x = \lim_{x \to 0} \frac{\log x}{\frac{1}{x}} = \lim_{x \to 0} \frac{\frac{1}{x}}{-\frac{1}{x^2}} = \lim_{x \to 0} -x = 0,$$

we know that the limit of the fourth term of the right side of $(**)$ exists at $x = 0$. Thus the definite integral $(*)$ exists, and we know after simple calculation that it has a value of $\frac{1}{8}$. The last component has the same value. In addition, we have the first- and second-order derivatives:

	xy	$(1-x)(1-y)$	$\frac{x^2(1-y)}{1+x-y}$	$\frac{(1-x)^2 y}{1-x+y}$
$\frac{\mathrm{d}}{\mathrm{d}x}$	y	$y-1$	$-\frac{x(y-1)(x-2y+2)}{(x-y+1)^2}$	$\frac{y(x-1)(2y-x+1)}{(y-x+1)^2}$
$\frac{\mathrm{d}}{\mathrm{d}y}$	x	$x-1$	$-\frac{x^3}{(x-y+1)^2}$	$-\frac{(x-1)^3}{(y-x+1)^2}$
$\frac{\partial^2}{\partial^2 x}$	0	0	$\frac{2(y-1)^3}{(x-y+1)^3}$	$\frac{2y^3}{(y-x+1)^3}$
$\frac{\partial^2}{\partial x \partial y}$	1	1	$-\frac{x^2(x-3y+3)}{(x-y+1)^3}$	$-\frac{(x-1)^2(3y-x+1)}{(y-x+1)^3}$
$\frac{\partial^2}{\partial y \partial x}$	1	1	$-\frac{x^2(x-3y+3)}{(x-y+1)^3}$	$-\frac{(x-1)^2(3y-x+1)}{(y-x+1)^3}$
$\frac{\partial^2}{\partial y \partial y}$	0	0	$-\frac{2x^3}{(x-y+1)^3}$	$\frac{2(x-1)^3}{(y-x+1)^3}$

The stationary point $\left(\frac{1}{2}, \frac{1}{2}\right)$ (with $\frac{\mathrm{d}p^{(1)}}{\mathrm{d}x}\left(\frac{1}{2}, \frac{1}{2}\right) = 0$ and $\frac{\mathrm{d}p^{(1)}}{\mathrm{d}y}\left(\frac{1}{2}, \frac{1}{2}\right) = 0$) induces $\frac{\partial^2}{\partial^2 x}\frac{\partial^2}{\partial^2 y} - \frac{\partial^2}{\partial y \partial x}\frac{\partial^2}{\partial x \partial y} < 0$. At this saddle point, $p^{(1)} = \frac{3}{4}$.

We notice that this is the case when in a particle's transition matrix there are only two nonzero items (see (3.6), (3.7)). If a particle has more possible choices (that is, more nonzero items in the transition matrix), the chance of a successful step execution in the simulation cycle will be substantially lower. Consequently, the expected average pedestrian walking speed $1.33\,\mathrm{m}\cdot\mathrm{s}^{-1}$ will not be reached.

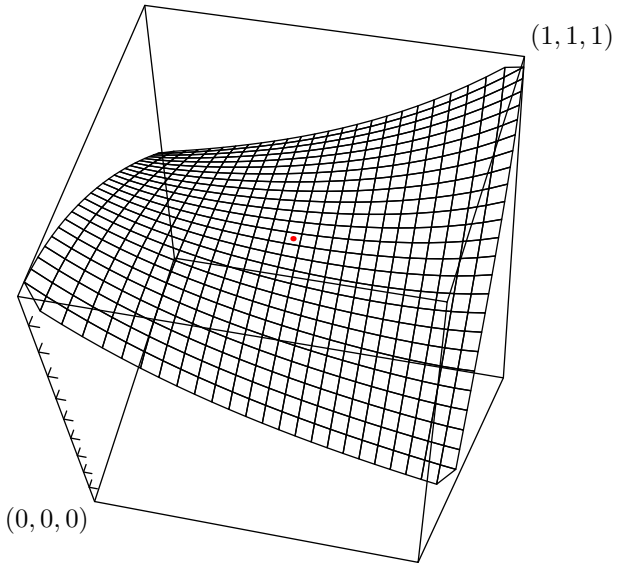

Figure 3.6: $p^{(1)}$ in the interval $(0,1)\times(0,1)$ shown as a surface in a three-dimensional frame. $p^{(1)}$ is not evaluated at $(p_1,p_2)\in\{(0,0),(0,1),(1,0),(1,1)\}$. The saddle point (drawn in red) at $\left(\frac{1}{2},\frac{1}{2}\right)$ has a value of $\frac{3}{4}$.

3.1.3 A local approach

For position transition we propose a completely different approach. First, the number of the elementary steps required for a certain position transition will be called the *topological* length of this transition. It is to be distinguished from the geometric length of this position transition.

3.1.3.1 Neutralized elementary step execution

The central idea of our method is to minimize the discrepancy between geometric and topological lengths. In the ideal case, geometric and topological lengths should be equal. This idea had been first presented in our previous work [10].

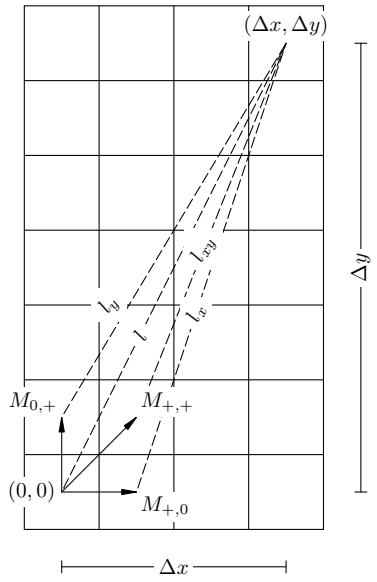

Figure 3.7: Neutralization of the diagonal elementary step. The elementary steps $M_{+,0}$, $M_{0,+}$ and $M_{+,+}$ are associated with probabilities p_x, p_y and p_{xy} respectively.

Without loss of generality, we request $\Delta x \geq 0$, $\Delta y > 0$ and thus the corresponding elementary steps are confined to be $M_{+,0}$, $M_{0,+}$ and $M_{+,+}$. We recall that these elementary steps refer to a move in the x-direction, a move in the y-direction and a move in both directions respectively. These possible elementary steps will be assigned probability numbers $p_x, p_y, p_{xy} \in [0,1]$ respecting their execution. Obviously, the first bounding condition concerning p_x, p_y and p_{xy} is:

$$p_x + p_y + p_{xy} = 1. \tag{3.8}$$

A second bounding condition can be given as:

$$p_x \cdot \Delta y = p_y \cdot \Delta x. \tag{3.9}$$

This requirement is to say that the likelihood an axis-parallel elementary step is to be carried out should correlate with the geometric lengths of the components Δx and Δy in the purposed position transition. In other words, we expect that the larger magnitude the position transition is in a certain dimension, the more in that direction an axis-parallel elementary step should be favoured. We see that the trivial case $\Delta x = 0$ induces $p_x = 0$. In this case, we let $p_{xy} = 0$ and get $p_y = 1$, this is to say that there is no need to make any move with an nonzero x-component, when we already have $\Delta x = 0$.

Next, we notice that the distance from $(\Delta x, \Delta y)$ to the current start position $(0,0)$ is:

$$l = \sqrt{(\Delta x)^2 + (\Delta y)^2}.$$

Similarly, from $(\Delta x, \Delta y)$, the distance l_x to the new position after an elementary step $M_{+,0}$, the distance l_y to the new position after an elementary step $M_{0,+}$ and the distance l_{xy} to the new position after an elementary step $M_{+,+}$ can be written explicitly as:

$$\begin{aligned} l_x &= \sqrt{(\Delta x - 1)^2 + (\Delta y)^2}, \\ l_y &= \sqrt{(\Delta x)^2 + (\Delta y - 1)^2}, \\ l_{xy} &= \sqrt{(\Delta x - 1)^2 + (\Delta y - 1)^2}, \end{aligned}$$

cf. Figure 3.7.

We request the third bounding condition for p_x, p_y and p_{xy} (in the nontrivial case $\Delta x > 0$, $\Delta y > 0$):

$$p_x(1 + l_x) + p_y(1 + l_y) + p_{xy}(1 + l_{xy}) = l. \tag{3.10}$$

We are under the assumption that, in the ideal case, the topological length, measured in the number of the required elementary steps, should be equal to the geometric length of the position transition. This is established by (3.10): after the execution of the corresponding elementary step defined by the current p_x, p_y and p_{xy}, the aforesaid assumption will remain valid with the updated rest lengths l_x, l_y and l_{xy}.

The solution of (3.8)–(3.10) is:

$$p_y = \frac{l - l_{xy} - 1}{l_y + \frac{\Delta x}{\Delta y} l_x - \left(1 + \frac{\Delta x}{\Delta y}\right) l_{xy}}, \quad p_x = \frac{\Delta x}{\Delta y} p_y, \quad p_{xy} = 1 - p_x - p_y. \tag{3.11}$$

Depending on the values of p_x, p_y and p_{xy}, an elementary step M can be performed.

Remark: Unlike Bresenham's algorithm, the method (3.11) computes the values of p_x, p_y and p_{xy} relating to a given configuration of position transition $(\Delta x, \Delta y)$. After the execution of the selected elementary step M, the rest position transition is to be modified. For the elementary step to follow, the computation of p_x, p_y and p_{xy} will be based on this new configuration of $(\Delta x, \Delta y)$. This process will be repeated until the purposed position transition is completely realized, in which case $\Delta x = \Delta y = 0$.

For negative Δx (and Δy), we only need to negate the moving direction in x (and that in y) in the elementary steps (that is, $M_{+,0}$ will be replaced by $M_{-,0}$ and so on). The calculation of the values of p_x, p_y and p_{xy} will not be affected, when we replace Δx (and Δy) by $|\Delta x|$ (and $|\Delta y|$).

Remark: We look into two special cases. In the trivial case $\Delta x = 0$, $\Delta y = 1$, (3.11) gives $p_y = 1$, $p_x = p_{xy} = 0$, which points to the rather unambiguous elementary step $M_{0,+}$ (axis-parallel in the y-direction). Another trivial case $\Delta x = \Delta y = 1$ provides us with $p_y = p_x = \frac{\sqrt{2}-1}{2}$, $p_{xy} = 2 - \sqrt{2}$, associated with this combination of p_x, p_y and p_{xy}, the mathematical expectation of the number of elementary steps needed for the realization of the position transition $(1, 1)$ is exactly $\sqrt{2}$.

(3.11) assumes that all the three elementary steps $M_{+,0}$, $M_{0,+}$ and $M_{+,+}$ are indeed possible. This may not always be the case. The situation of exactly one possible elementary step or no elementary step is possible at the moment is self-evident. In general, however, an unavailable elementary step choice will be assigned an execution probability of zero. If the diagonal elementary step choice $M_{+,+}$ is unavailable, by setting $p_{xy} = 0$, p_x and p_y can be deduced from (3.8)

and (3.9), (3.10) will not be needed.

If an axis-parallel elementary step choice is not available, let this without loss of generality be $M_{0,+}$, we can modify (3.8) and (3.10) into:

$$\begin{aligned} p_x \qquad\quad &+ p_{xy} \qquad\quad &= 1, \\ p_x(1+l_x) &+ p_{xy}(1+l_{xy}) &= l. \end{aligned}$$

The solution of the new bounding conditions now becomes:

$$p_x = \frac{l - l_{xy} - 1}{l_x - l_{xy}}, \quad p_{xy} = \frac{l_x - l + 1}{l_x - l_{xy}}. \tag{3.12}$$

In Tables 3.1 and 3.2, the solution (3.11) and (3.12) is given in approximation for some common combinations of Δx and Δy. Referring to Table 3.1, the probability $p_{xy} = 1 - p_x - p_y$ can be easily retrieved from the accumulated probability $p_x + p_y$. Table 3.2 refers to $p_{xy} = 1 - p_x$.

Table 3.1: Look-up table for the solution by (3.11), in terms of accumulated probabilities p_x and $p_x + p_y$, for $\Delta x, \Delta y = 0, \ldots, 7$, $(\Delta x, \Delta y) \neq (0, 0)$.

	$\Delta x = 0$	$\Delta x = 1$	$\Delta x = 2$	$\Delta x = 3$	$\Delta x = 4$	$\Delta x = 5$	$\Delta x = 6$	$\Delta x = 7$
$\Delta y = 0$		1 1	1 1	1 1	1 1	1 1	1 1	1 1
$\Delta y = 1$	0 1	0.207 0.414	0.258 0.387	0.285 0.380	0.299 0.373	0.306 0.368	0.312 0.363	0.315 0.360
$\Delta y = 2$	0 1	0.129 0.387	0.252 0.504	0.305 0.509	0.335 0.503	0.354 0.496	0.367 0.489	0.376 0.484
$\Delta y = 3$	0 1	0.095 0.380	0.204 0.509	0.267 0.533	0.306 0.536	0.334 0.534	0.353 0.529	0.367 0.525
$\Delta y = 4$	0 1	0.075 0.373	0.168 0.503	0.230 0.536	0.273 0.547	0.305 0.549	0.329 0.548	0.347 0.545
$\Delta y = 5$	0 1	0.061 0.368	0.142 0.496	0.200 0.534	0.244 0.549	0.278 0.555	0.304 0.556	0.324 0.556
$\Delta y = 6$	0 1	0.052 0.363	0.122 0.490	0.176 0.529	0.219 0.548	0.253 0.556	0.280 0.560	0.302 0.561
$\Delta y = 7$	0 1	0.045 0.360	0.108 0.484	0.157 0.525	0.198 0.545	0.231 0.556	0.259 0.561	0.282 0.564

Table 3.2: Look-up table for the solution p_x by (3.12), for $\Delta x, \Delta y = 0, \ldots, 7$, $(\Delta x, \Delta y) \neq (0, 0)$.

	$\Delta x = 0$	$\Delta x = 1$	$\Delta x = 2$	$\Delta x = 3$	$\Delta x = 4$	$\Delta x = 5$	$\Delta x = 6$	$\Delta x = 7$
$\Delta y = 0$		1	1	1	1	1	1	1
$\Delta y = 1$	0	0.414	0.570	0.687	0.759	0.804	0.836	0.859
$\Delta y = 2$	0	0.236	0.504	0.624	0.699	0.751	0.788	0.816
$\Delta y = 3$	0	0.162	0.399	0.533	0.619	0.680	0.725	0.759
$\Delta y = 4$	0	0.123	0.322	0.455	0.547	0.614	0.664	0.704
$\Delta y = 5$	0	0.099	0.269	0.393	0.485	0.555	0.610	0.653
$\Delta y = 6$	0	0.083	0.229	0.344	0.433	0.504	0.560	0.606
$\Delta y = 7$	0	0.071	0.200	0.305	0.390	0.460	0.517	0.564

Technical note. In software implementation, Tables 3.1 and (3.2) would refer to a header file (or otherwise an independent program definition), which should be machine-generated to avoid input error.

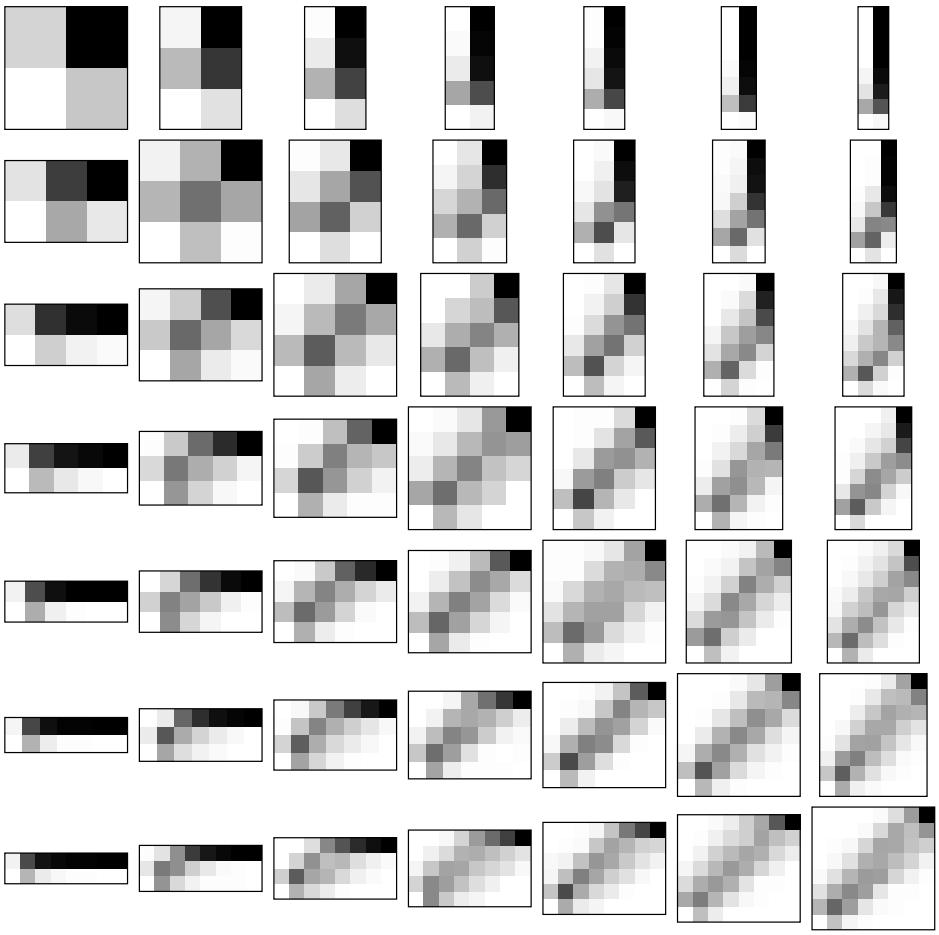

Figure 3.8: Occurrence of the grid cell position after 100 probes of elementary step execution for $(\Delta x, \Delta y)$ with $\Delta x, \Delta y = 1, \ldots, 7$. The relative occurrence of the cell position averaged by the number of total probes is shown in a linear gray scale. The cell position $(\Delta x, \Delta y)$ is always drawn in black since it represents the end position in the sequence of elementary step execution. The start position $(0, 0)$ is not counted and hence always in white. Different "trail" patterns can be clearly identified.

Figure 3.8 shows the grid cell position's occurrence in the execution of possible elementary steps by (3.11). For each configuration of $(\Delta x, \Delta y)$, one hundred probes had been undertaken. In each of these probes, a sequence of elementary steps from $(0, 0)$ to $(\Delta x, \Delta y)$ was constructed by (3.11) on a random basis and free of disturbance. The latter is to say that all the grid cell positions had always been free and accessible for the potential elementary step execution. For each configuration of $(\Delta x, \Delta y)$, the formation of a "trail" corresponding to the position

transition can be identified.

With consideration of this, deviations en route within a local step (i. e. within a simulation cycle which could take a time span of one to a couple of seconds) would become possible and strictly speaking, unavoidable. On the other hand, these local deviations are all under the restriction that the mathematical expectation of the sum of such elementary steps (of which the position transition in a simulation cycle is composed) should be equal to the position evolution in geometric sense. Therefore, this is exactly what we need in the modelling. Furthermore, the notion of the elementary step execution enables the representation of heterogeneous pedestrians.

3.1.3.2 Balancing mechanism of collected step execution

Pedestrians in a simulation system are often called particles. It is obvious that these particles often do not share with each other the same physical characteristics, among which velocity is the most important one. In a simulation system in discrete space, we will have heterogeneous particles with different (geometric and topological) step lengths in the simulation cycle. In each simulation cycle, the sequences of the elementary steps of the particles are to be decided. To this end, we introduce a balancing mechanism in the simulation cycle for the particles. Along with the elementary step calculation for a single particle proposed in the previous passage, this method provides a rather good solution for the simulation of a collection of particles on the operational level.

For a single particle, the execution of an elementary step in a simulation cycle can be with or free of disturbance. By "free of disturbance" we refer to the situation that all the grid cell positions involved in a potential elementary step execution are free and accessible for the current particle. In this case, the solution (3.11) will be applied in the calculation. Otherwise, we need either both (3.8) and (3.9), if the diagonal move is not possible, or (3.12), if an axis-parallel move is not available. Now we consider the disturbed case with potential conflicts with other particles.

Let $v_{\max}$ denote the highest possible speed of all the particles, $v_{\max}$ can be understood as the largest possible topological step length in the simulation cycle as well. Let $n \geq v_{\max}$ ($n \in \mathbb{N}$). It is possible to divide the simulation cycle into n equal intervals. As a matter of fact, a particle of speed v should perform exactly v elementary steps in these n intervals. To decide in which intervals the elementary steps will be performed, it is possible to choose one of the $\frac{n!}{v!\cdot(n-v)!}$ combinations from the set $\{0, \ldots, n-1\}$. Alternatively, we can select v numbers from the set $\{0, \ldots, n-1\}$ at equal (or similar) distances as indices to signal the intervals in which a particle is to be active to carry out the elementary steps.

However, in both cases possible conflicts among the particles are not dealt with properly.

The target position in correspondence of an elementary step may not always be available (due to the presence of other particles, for example). Therefore it can happen as well that no possible elementary step is available for the current particle. For this purpose, we calculate, in each of the n intervals of the simulation cycle a probability number for the particle to decide whether an elementary step is to be performed at the current moment. For a position transition of geometric length l (l is not limited to be an integer), in the ideal case, l elementary steps should be performed. Therefore, the likelihood to carry out an elementary step depends on how successfully the elementary steps in the current simulation cycle have already been realized. This idea is stated in the following procedure:

```
   Data: global information
   Input: particle (each with a local variable a), j
 1 if position transition resolved then
 2 |   mark particle as processed;
 3 else
 4 |   perform an elementary step with probability min((v−a)/j, 1);
 5 |   if elementary step execution successful then
 6 |   |   a ⇐ a + 1;
 7 |   end
 8 end
```

Procedure OperateSingle(particle, j)

In the above program fragment, the parameter j serves as an input time index. a is a local (member) variable of the particle, it records the number of successful elementary steps in the current simulation cycle, it is initialized to be 0 (cf. the next code fragment as caller procedure). v is the discrete speed of the particle, which is equal to the geometric length of the planned position transition within one simulation cycle. It can be seen that unsuccessful elementary steps in the current simulation cycle contribute to a higher probability that an elementary step should be attempted the next time (since v and a stay unchanged but j is decremented, cf. the next code fragment). The next code fragment describes the operation of all the particles in the current simulation cycle. Time index j will be applied to all the particles in the n operation intervals of a simulation cycle.

On the operational level, potential conflicts among the particles can be substantially reduced by Procedure OperateAll. Here the time index j is initialized to be n (that is, the number of the intervals in the simulation cycle) and will be decremented exactly n times; every time, Procedure OperateSingle will be called for each of the particles. In plain words, this procedure guarantees all particles in the simulation cycle a fair chance to perform their elementary steps.

Data: global information
initialize all particles;
initialize for each particle $a \Leftarrow 0$;
$j \Leftarrow n$;
while $j \geq 1$ **do**
  call OperateSingle(p, j) for every particle p not yet processed;
  $j \Leftarrow j - 1$;
end

Procedure OperateAll

Discussion about the tactical level will be continued in § 3.1.5. Conflict solution on the strategic level (or even higher) is a topic concerning routing and description of human behaviour in a social-psychological context, and is not included in the current research.

3.1.4 Continuous density

Here we will propose a simple navigation method for the particles based on continuous density estimation to model the interaction among them. We consider this a possible solution on the tactical level. We first explain the idea of continuous density.

In an observation area $\Omega \in \mathbb{R}^2$, a given number of n ($n > 0$) particles leads to a global density of $\rho = \frac{n}{|\Omega|}$ (where $|\cdot|$ denotes the size of an area). In a given neighbourhood Ω_0 respecting a position $\mathbf{x} \in \Omega$, the discrete local density would be:

$$\rho(\mathbf{x}) = \frac{n_0(\mathbf{x})}{|\Omega_0(\mathbf{x})|}\,, \tag{3.13}$$

where $n_0(\cdot)$ denotes the number of the particles in the neighbourhood of a position. However, the smoothness of the quantity ρ relies heavily on the choices of Ω_0. [68] proposed a density estimator based on the Voronoi decomposition in which the Voronoi polygons associated with the particle distribution will be used as local observation areas. [26] gave another solution:

$$\rho(\mathbf{x}, t) = \frac{1}{\pi R^2} \sum_{i=0}^{n-1} e^{-\frac{|\mathbf{x}_i(t) - \mathbf{x}|_2^2}{R^2}}\,, \tag{3.14}$$

where $\mathbf{x}_i$ is the position of the particle indexed by i and $|\cdot|_2$ refers to the Euclidean distance. (3.14) can be seen as a Gaussian kernel estimator with a fixed bandwidth. In this sense, each particle i is associated with a circle with origin $\mathbf{x}_i$ and radius R. Owing to the exponential function, the distance of an arbi-

trary position $\mathbf{x}$ to this circle will have a significant effect on the evaluation of $\rho(\mathbf{x}, t)$; on the other hand, this associated circle resembles the personal space of a particle—as an abstraction of a real pedestrian—which it reserves for itself exclusively. Empirically, however, pedestrian personal spaces are not of a unanimous size.

Our previous work [54] introduced a modified estimator based on the nearest neighbour method (see [65], page 19):

$$\hat{\rho}_I(\mathbf{x}, t) = \frac{1}{\pi} \sum_{i \in I} \frac{1}{R_i^2} \cdot e^{-\frac{|\mathbf{x}_i(t) - \mathbf{x}|_2^2}{R_i^2}} , \tag{3.15}$$

with $R_i = \lambda d_i$, where $\lambda > 0$ is a smoothing parameter and d_i is the distance from particle i to the nearest obstacle—including especially, the other particles—in the whole observation area, and $I = \{0, \ldots, n-1\}$ is the index set of the particles. If it is clear in the context, the index set I as subscript can be left out.

We consider the case $R_i \to 0$ (this is when particles have infinitesimally small sizes). By (3.15) we have:

$$\frac{1}{|\Omega_0(\mathbf{x})|} \int_{\Omega_0(\mathbf{x})} \hat{\rho}(\mathbf{x}, t) \, \mathrm{d}S = \frac{1}{|\Omega_0(\mathbf{x})|} \sum_{i \in I} \frac{1}{\pi R_i^2} \int_{\Omega_0(\mathbf{x})} e^{-\frac{|\mathbf{x}_i(t) - \mathbf{x}|_2^2}{R_i^2}} \, \mathrm{d}S$$

by approximation of the Dirac δ-function in two dimensions,

$$\begin{aligned} &\doteq \frac{1}{|\Omega_0(\mathbf{x})|} \sum_{i \in I} \int_{\Omega_0(\mathbf{x})} \delta(\mathbf{x}_i(t) - \mathbf{x}) \, \mathrm{d}S \\ &= \frac{n_0(\mathbf{x}, t)}{|\Omega_0(\mathbf{x})|} , \end{aligned}$$

the latter would be the discrete local density of position $\mathbf{x}$ at time t, cf. (3.13).

Remark: The definition (3.15) can be applied to subsets of the whole particle group, in which case d_i refers to the distance from a particle i to its nearest obstacle including all the other particles in the whole observation area instead of those from the particular subset. For an arbitrary partition J_k ($k = 0, \ldots, m-1$, $m \geq 1$) of the index set I associated with the particles (that is, $J_k \neq \emptyset$, for $k = 0, \ldots, m-1$; $J_{k_1} \cap J_{k_2} = \emptyset$, for $k_1, k_2 = 0, \ldots, m-1$, $k_1 \neq k_2$; and $\cup_k J_k = I$) in the observation area, we have $\hat{\rho}_I(\mathbf{x}, t) = \sum_k \hat{\rho}_{J_k}(\mathbf{x}, t)$. The partition of the index set I can be seen as a group separation of the original particles with which the sum of the group densities renders the total density in return.

3.1.5 Local navigation based on continuous density

We apply the continuous density $\hat{\rho}$ on positions in discrete geometry, that is, density on discrete positions will be evaluated[5] by $\hat{\rho}$. On a fully populated grid (that is, in each grid cell there is exactly one particle), we have $R_i = \lambda$ for all $i \in I$. If we set $\lambda = \sqrt{2}$, $\hat{\rho}(\mathbf{x}, t)$ becomes very close to 1 everywhere on the grid, when the latter grows to be infinite ($n_x \to +\infty$, $n_y \to +\infty$);[6] this is exactly what we would expect for local density.

We write particle position $\mathbf{x}_i = (x_i, y_i)$ in discrete form; accordingly, (3.15) can be rewritten as

$$\hat{\rho}_I(x, y) = \frac{1}{\pi} \sum_{i \in I} \frac{1}{2d_i^2} \cdot e^{-\frac{(x_i - x)^2 + (y_i - y)^2}{2d_i^2}} , \tag{3.16}$$

parameter t for time has been left out for simplicity's sake.

An example of the continuous density on square grid is given in Figure 3.9. It can be concluded that a single isolated particle i does not induce high density locally, since in this case its distance to the nearest obstacle d_i is larger than that of a particle which is closer to other particles or obstacles. For example, in Figure 3.9, the continuous density at position $(4, 4)$ is 0.026, whereas at $(6, 6)$ it has value 0.172, since the particle at $(6, 6)$ is next to the boundary and the corresponding d_i for the latter is 1. Figure 3.10 illustrates the same situation of Figure 3.9 with periodic horizontal boundaries, that is, bordering positions $(0, 0), \ldots, (0, n_y - 1)$ will be connected with $(n_x - 1, 0), \ldots, (n_x - 1, n_y - 1)$ respectively. We see in Figure 3.10 that particles at positions $(0, 2)$ and $(0, 3)$ are immediate neighbours to particles at $(13, 2)$ and $(13, 3)$, and this leads to higher densities locally.

We have noticed that the particle index set I can be partitioned into index sets of different groups. Let $A, B, \ldots$ denote such particle groups. Let the continuous density at a discrete position (x, y) induced by the particles of an arbitrary

[5] Distances will be discrete as well.

[6] We give an estimate for $\sum_{x=-\infty}^{+\infty} \sum_{y=-\infty}^{+\infty} e^{-\frac{x^2+y^2}{2}}$; in fact, this sum is the square of $\sum_{x=-\infty}^{+\infty} e^{-\frac{x^2}{2}}$. Let X be a positive integer. We have

$$\sum_{|x|>X} e^{-\frac{x^2}{2}} < 2 \int_X^{+\infty} e^{-\frac{x^2}{2}} \, \mathrm{d}x < \frac{2}{X} \int_X^{+\infty} x e^{-\frac{x^2}{2}} \, \mathrm{d}x = \frac{2}{X} e^{-\frac{X^2}{2}} .$$

Hence, by

$$\sum_{x=-X}^{X} e^{-\frac{x^2}{2}} < \sum_{x=-\infty}^{+\infty} e^{-\frac{x^2}{2}} < \sum_{x=-X}^{X} e^{-\frac{x^2}{2}} + \frac{2}{X} e^{-\frac{X^2}{2}} ,$$

upper and lower bounds of the sum $\sum_{x=-\infty}^{+\infty} e^{-\frac{x^2}{2}}$ can be found. Choosing $X = 4$, we will have $2.5066208 < \sum_{x=-\infty}^{+\infty} e^{-\frac{x^2}{2}} < 2.5067886$, whereas $\sqrt{2\pi} = 2.50662827 \cdots$.

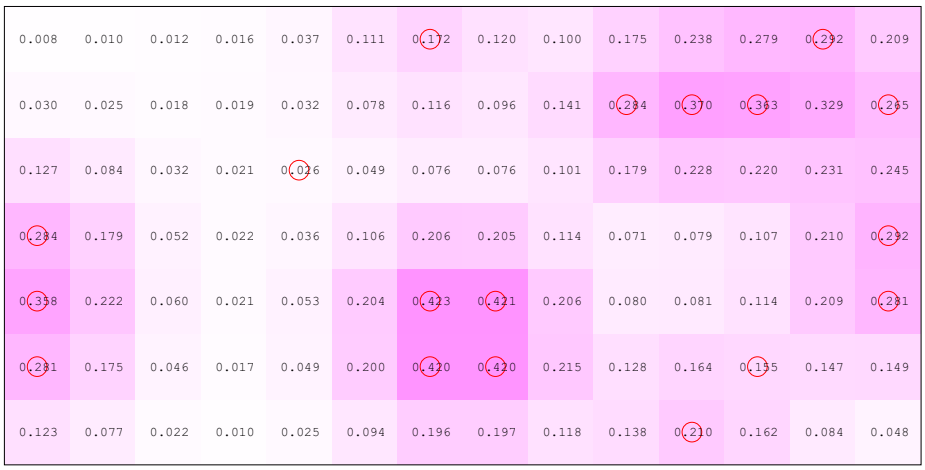

Figure 3.9: Example of continuous density in numerical values on a square grid. All particles (shown in red circles) belong to the same group. The outer boundary of the square grid (that is, positions $(-1,-1),\ldots,(n_x,-1),\ldots,(n_x,n_y),\ldots,(-1,n_y),\ldots,(-1,0))$ is considered as obstacles in the evaluation of d_i.

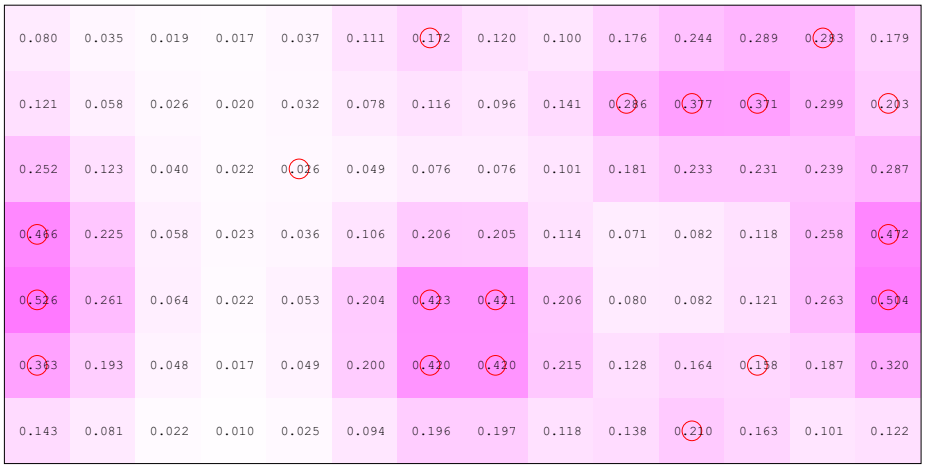

Figure 3.10: Example of continuous density in numerical values on a square grid with periodic horizontal boundary with the same particle distribution as in Figure 3.9.

group A be written as $\hat{\rho}_A(x,y)$. Let $\bar{A}$ denote the complement of A, density at position (x,y) induced by those particles not belonging group A will be written as $\hat{\rho}_{\bar{A}}(x,y)$. In case of a simple partition of I by two groups A and B, we have $\bar{A}=B$ and we write $\hat{\rho}_{\bar{A}}=\hat{\rho}_B$. Empirical observations show that pedestrians tend to follow their own group while avoiding members of a foreign group (or

foreign groups). Under this assumption, we define:

$$o_A(x, y) = \hat{\rho}_{\bar{A}}(x, y) - \hat{\rho}_A(x, y), \tag{3.17}$$

o_A will be called *density overlay* or simply *overlay*. A grid cell position (x, y) with a small value of o_A is considered to be "attractive" for particles of group A. The grid cell position with the lowest overlay can be chosen as a temporary destination for the current particle on the tactical level.

(3.17) shows a compensation effect: the attraction incurred by one's own group can be counterbalanced by the foreign group (or groups). This compensation can be of disadvantage when high densities are involved. A very high local density indicates that the personal space of a pedestrian (see discussion on page 46) may be infringed. In such a case, compensation of group densities can give a wrong signal. We introduce two empirical threshold values for group-induced density θ_{self} and θ_{other}. Respecting particles from an arbitrary group A, we write $\theta_A = \theta_{\text{self}}$ and $\theta_{\bar{A}} = \theta_{\text{other}}$. A revised overlay can be defined as

$$o'_A(x, y) = \begin{cases} 1, & \text{if } \hat{\rho}_{\bar{A}}(x, y) > \theta_{\bar{A}}, \\ \hat{\rho}_{\bar{A}}(x, y) + \hat{\rho}_A(x, y), & \text{if } \hat{\rho}_{\bar{A}}(x, y) \leq \theta_{\bar{A}} \text{ and } \hat{\rho}_A(x, y) > \theta_A, \\ o_A(x, y), & \text{otherwise.} \end{cases} \tag{3.18}$$

In the first case of (3.18), the induced density of the foreign group is above the corresponding threshold value, by setting the overlay to be 1 (which is, in approximation, the maximum possible density on a grid, see discussion on page 60) we make the current grid position definitely unfavourable for any potential incoming pedestrians. If otherwise the induced density of the own group $\hat{\rho}_A(x, y)$ is above the corresponding threshold value, it will be considered as a negative element in evaluating the overlay. Similar to the first case, the current grid position will be unfavourable for other particles. In fact, (3.17) is a special case of (3.18) (setting $\theta_A = \theta_{\bar{A}} = 1$).

Remark: To some extent, the second case of (3.18) can be confirmed by an assumption made in [30] regarding multiple types (groups) of pedestrians: "Hypothesis 1A. The speed of pedestrians of a single type in multiple type flow is still determined by the function $f(\rho)$ but where ρ is now the total density rather than the density of a single pedestrian type" (section 4).

Remark: Although $o'_A(x, y)$ is not a continuous function, it reflects the exclusiveness of the pedestrians' personal space, since the need (or tendency) for a pedestrian to repulse other pedestrians diminishes significantly outside this area.

3.1.6 Simulation and model calibration

In this section we give a detailed description of our model composition and possible calibration. Our task is to model the interaction dynamics among pedestrian groups.

3.1.6.1 Basic configuration

For the system geometry we adopt the common configuration of a grid cell size of $0.4\,\mathrm{m} \cdot 0.4\,\mathrm{m}$ and an average pedestrian walking speed of $1.33\,\mathrm{m} \cdot \mathrm{s}^{-1}$ (cf. § 3.1.2). This grid cell size would be the exclusive personal space of the pedestrians. The average speed in discrete form becomes $\bar{v} = 3.325$. On the other hand, we need to request $v_{\min} = 1$, so that in discrete space, the position transition associated with a particle's default speed is not smaller than the one-dimensional grid size[7]. In addition, we assume that the pedestrian's default speed obeys a normal distribution. However, in reality the speed distribution is not so clear as it should look like. [3] reported an average walking speed of $0.9\,\mathrm{m} \cdot \mathrm{s}^{-1}$, for people aged between 65 and 69 years, with a standard deviation of $0.3\,\mathrm{m} \cdot \mathrm{s}^{-1}$, and $0.8\,\mathrm{m} \cdot \mathrm{s}^{-1}$ for people aged 65 years and above, with a standard deviation roughly the same. [56] collected various results from previous studies and presented an average walking speed of $1.32543\,\mathrm{m} \cdot \mathrm{s}^{-1}$ with a standard deviation of $0.183291\,\mathrm{m} \cdot \mathrm{s}^{-1}$.[8]

In our simulation, we consider the normal distribution $X \sim \mathcal{N}(0, \sigma^2)$, the speed of the particles can be defined by a valid X via:

$$v = \bar{v} + (\bar{v} - v_{\min}) \cdot X, \tag{3.19}$$

if the random variable X is not within the range $[-1, 1]$, it will be discarded. With $\sigma = \frac{1}{3}$, appropriately 99.7% generated values will be valid, with $\sigma = \frac{1}{4}$, more than 99.99%. Translated back into continuous form, the standard deviation would be $((1.33 - 0.4) \cdot \sigma)\ \mathrm{m} \cdot \mathrm{s}^{-1}$; with $\sigma = \frac{1}{3}$, we have a standard deviation of $0.31\,\mathrm{m} \cdot \mathrm{s}^{-1}$, and for $\sigma = \frac{1}{4}$, it becomes $0.2325\,\mathrm{m} \cdot \mathrm{s}^{-1}$. The speed v generated by (3.19) will be the individual default speed of the particle, that is, the discrete length this particle can cover on the grid during one simulation cycle under undisturbed circumstances. The maximum possible speed is $\frac{2\bar{v} - v_{\min}}{\bar{v}} \cdot 1.33\,\mathrm{m} \cdot \mathrm{s}^{-1} = 2.26\,\mathrm{m} \cdot \mathrm{s}^{-1}$.

3.1.6.2 Model composition and system dynamics

The system dynamics are defined by the collected state evolution of the particles in the simulation system. On the tactical level, the simulation cycle is set to have a time length of one second. The choice of such a time length enables the

[7] Otherwise the discrete position of this particle will stay unchanged in the simulation.

[8] These data had been collected from some fifty events with no further indication of how many pedestrians were really engaged, so the validity of six significant digits might be questionable.

particles to make or adapt their decisions on the tactical level in a way similar to real pedestrians. Within a simulation cycle, interaction among the particles (pedestrians) is considered to be on the operational level.

Each particle will be given a global walking direction (pointing to an exit etc.). In each simulation cycle, similar to the cognitive modelling in § 1.3.4, every particle searches for a grid cell position within a perception area defined by a visibility angle ϕ from its current position at its default speed, see Figure 3.11 for an illustration. A grid position for the next position transition will be selected after comparing the corresponding overlay values. The visibility angle ϕ has been defined to be 75°. The size of ϕ has been chosen with consideration of real pedestrians' possible change of their walking directions within a relatively short time span of one second (that is, the length of a simulation cycle) to adapt themselves to the new environment. No further navigation method will be applied.

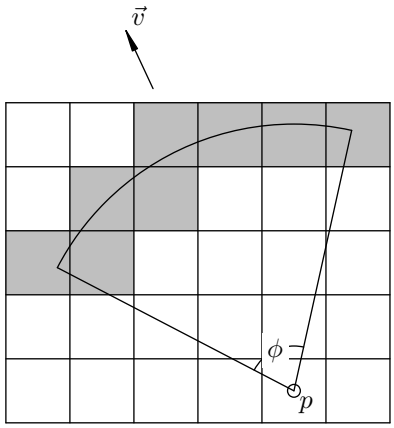

Figure 3.11: Within its visibility angle $\phi = 75°$, particle p on the tactical level faces the choice of a proper grid cell position by density overlay.

On the operational level, each particle will carry out a series of elementary steps through which its local position transition should be realized. The execution of the elementary steps has already been detailed in § 3.1.3.1 and § 3.1.3.2. The balancing mechanism ensures all particles equal chances in carrying out their elementary steps in the simulation cycle when local conflicts among them are inevitable.

3.1.6.3 Simulation results and calibration

During the *Lange Nacht der Wissenschaften*[9] (German: Long Night of the Sciences) *2010* a couple of experiments had been conducted in the Mathematics Department building of Technische Universität Berlin. The experiments had

[9]Homepage `https://www.langenachtderwissenschaften.de`.

been video-recorded to show the interaction among pedestrians in groups. In the experiments, participants were given instructions to walk in a certain direction (toward an exit etc.). Local position conflicts were solved by participating pedestrians themselves without guidance or interference from the experiment organizer.

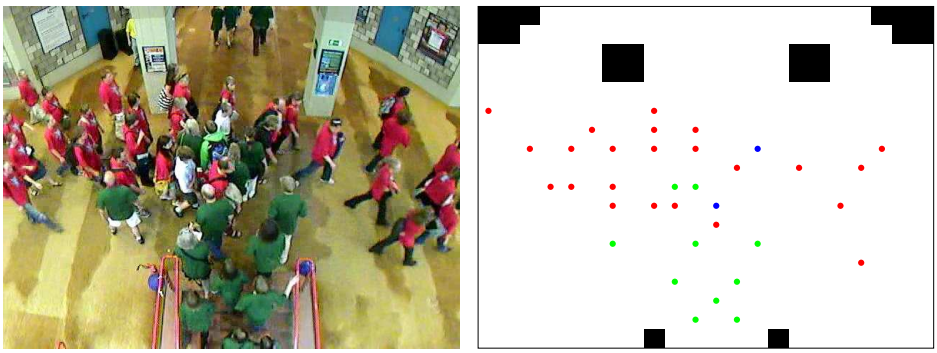

Figure 3.12: Experiment 1 from *LNdW 2010*. Left: Frame from the original video recording showing the scene ten seconds after the start. Right: The corresponding discrete representation on a square grid. The group in red was requested to walk from the left to the right side; the group in green was requested to walk to the upper side. Obstacles are shown in black blocks. The square grid has a size of $n_x = 22, n_y = 18$ and $l_c = 0.4\,\mathrm{m}$. The blue points in the right subfigure refer to the rather rare situation in which two (or more) sample points generated by the photogrammetric calculation happen to be discretized into the same grid cell position.

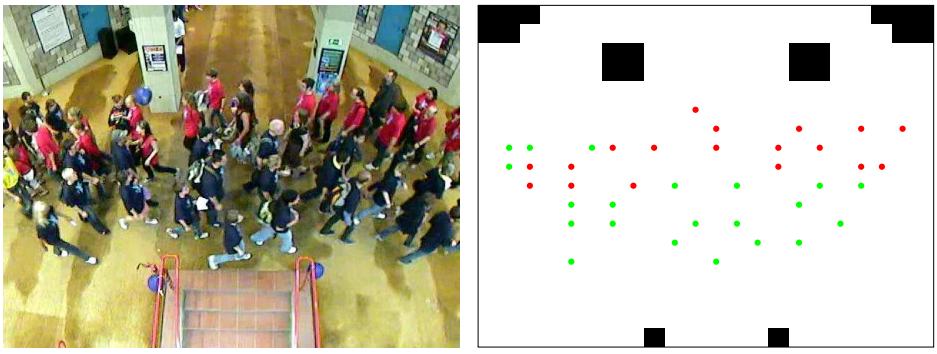

Figure 3.13: Experiment 2. Left: Frame from the original video recording showing the scene ten seconds after the start. Right: The corresponding discrete representation. The group in red was requested to walk from the right to the left side; the group in green walked in the opposite direction.

By means of photogrammetric calculation (see § 4.1 for details) we were able to retrieve the pedestrians' physical positions. These raw data will be transformed into discrete form, given a pre-defined grid cell size l_c. Video frame examples and

their discrete representations are given in Figures 3.12 and 3.13 which share the same background geometry.

Technical note. With a one-dimensional grid cell size of $l_c = 0.4\,\mathrm{m}$, we notice that in discrete space approximately 1.3% of the sample points (shown as blue points in Figure 3.12) in the whole sequence would come into conflict with each other. This is because the minimum net distance[10] between two particles can be smaller than the grid cell size l_c, although the particles' exclusive spaces are not really violated. By reducing l_c to 0.333 m, there will be about 0.6% of the sample points in conflict with each other. With a further reduction to 0.2 m, it still concerns about 0.1% of the particles. In our opinion, there is no particular need to downsize the grid cell size owing to this consideration.

To simulate the two scenarios, particles are generated with a combination of the default speed defined by (3.19) and the initial physical position with corresponding time index retrieved from the original video data. No further physical quantities of the original data will be implemented. Owing to the limited volume of the video recordings—in terms of time length and the number of pedestrians contained therein—it would be impossible to produce statistically convincing fundamental diagrams from the available input data. For the purpose of model validation, however, information about the time lapse of each pedestrian's passing the observation area has been collected. In Figure 3.14 we show the distribution of this time lapse (measured in seconds on the horizontal axis) in histograms after one hundred test runs of our simulation. Results generated from the original video recordings are printed in dark and light gray colours (referring to the two pedestrian groups), the corresponding simulation results are in red and green with the same colouring as in Figures 3.12 and 3.13. The actual time lengths of the passing are scenario-relevant only.

[10] By "net" we mean the distance between two particles deducted by their physical sizes. Applying the notation of (1.16), it can be written as $d_{ij} - r_i - r_j$.

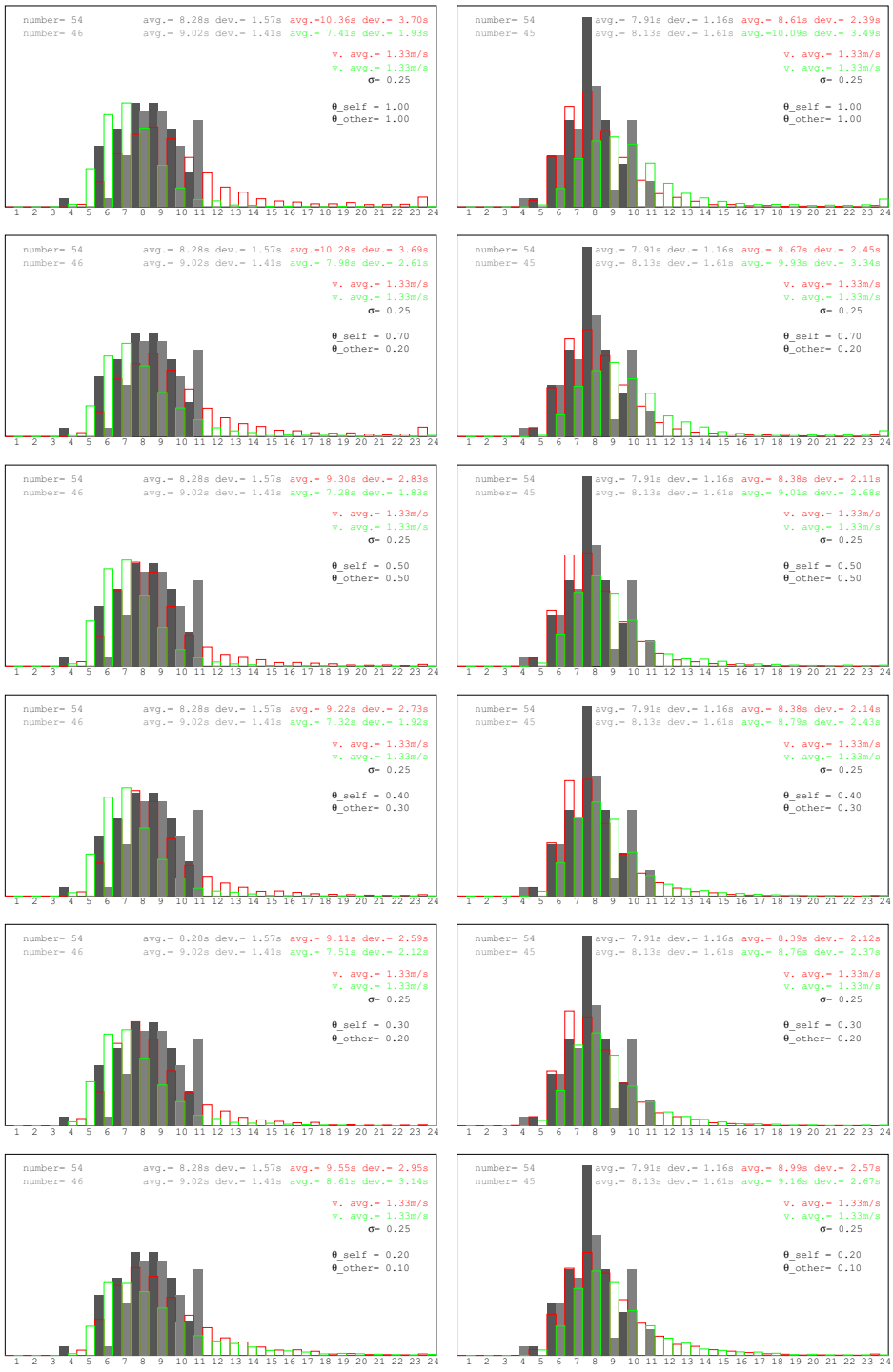

Figure 3.14: Simulation results of the two scenarios respecting the original experiments with various settings of θ_{self} and θ_{other}. Left column: Test scenario 1. Right column: Test scenario 2. Horizontal axis: Elapsed time lengths (in seconds). Vertical axis: Occurrence of the elapsed time lengths, shown in a relative linear scale.

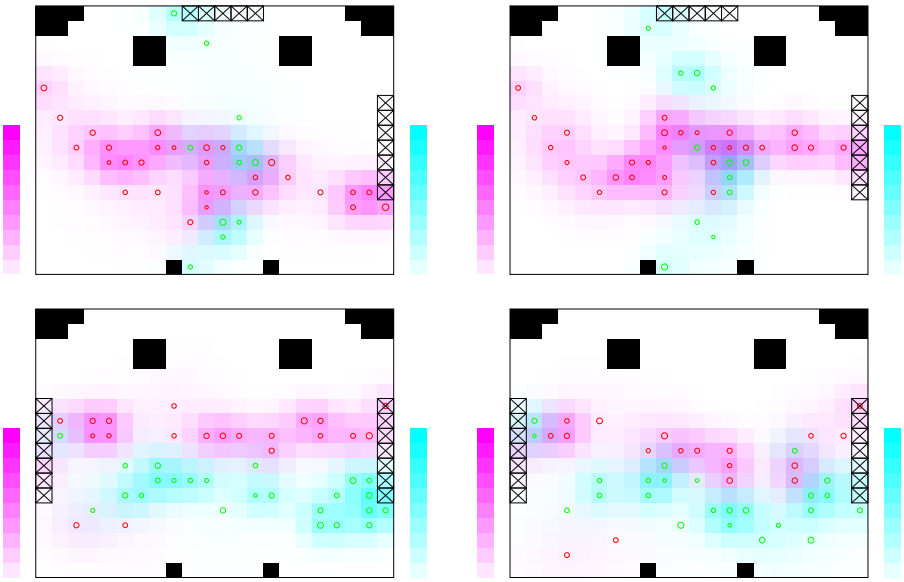

Figure 3.15: Test screenshots after ten seconds for comparison with the original data. First row: Test scenario 1. Second row: Test scenario 2. Left column: $(\theta_{\text{self}}, \theta_{\text{other}}) = (0.5, 0.5)$. Right column: $(\theta_{\text{self}}, \theta_{\text{other}}) = (0.3, 0.2)$. Exits are shown in "⊠"s.

Since particles tend to be less defensive toward members of their own groups, it is natural to request $\theta_{\text{self}} \geq \theta_{\text{other}}$. We present simulation results of six different combinations of θ_{self} and θ_{other}. We see in the first two rows of Figure 3.14 ($(\theta_{\text{self}}, \theta_{\text{other}}) = (1, 1)$ and $(\theta_{\text{self}}, \theta_{\text{other}}) = (0.7, 0.2)$) that a high θ_{self} does not produce good results: particle speeds are lower (that is, longer time lapses for passing) than expected (this is specially the case with the second scenario, see the subfigures in the right column there). This is caused[11] by the unfavourable compensation of the group densities (that is, the third case of (3.18) which further refers to (3.17)). We also notice that some of the particles need unreasonably much time to pass the observation area (see the lower right corners of the subfigures in the first two rows). In the last row, the setting $(\theta_{\text{self}}, \theta_{\text{other}}) = (0.2, 0.1)$ renders low particle speeds as well: owing to the low threshold values, overlay value can be 1 very often and thus invalidates potential candidates of position transition for the particles and consequently temporary route choices on the tactical level can be wrong.

The other three settings show similar but better results. Apart from the second group (that is, in green) in the first scenario, particles take a little more time[12] in the simulation tests to pass the observation area. In the data evaluation

[11] To be more exact, this is when the first case of (3.18) does not occur. However, The first case of (3.18) gives an overlay value 1, which often invalidates a position as candidate for temporary route choices.

[12] It is interesting to mention that in the first test scenario, particles from the second group (coloured in green) had a shorter distance to cover, despite this, they showed a larger time lapse (9.02 s in average) to pass the observation area. The reason for this is unknown. We wish that

of the original video recordings, the pedestrian's passing time is defined as the time lapse from the first appearance to the same pedestrian's last appearance in the observation area in the video sequence. However, given this time length, the pedestrian is still in the observation area, in other words, the pedestrian's real passing time should be slightly longer. With consideration of this, we may claim that our simulation produces rather good results.

In Figure 3.15 we present four screenshots of some random simulation tests applying the parameter settings $(\theta_{\text{self}}, \theta_{\text{other}}) = (0.5, 0.5)$ and $(0.3, 0.2)$. The screenshots were taken exactly ten seconds after the start (that is, after ten simulation cycles) for a possible comparison with the original video frames (cf. Figures 3.12 and 3.13). The grid cells have been coloured showing the induced group densities.

Remark: It is possible to calibrate the threshold values of θ_{self} and θ_{other} by brute force. All we need is to enumerate the possible patterns of grid cell position occupancies induced by the particles and calculate the continuous density under such circumstances. If we can define the should-be pedestrian behaviour in case of potential conflict with others, feasible θ_{self} and θ_{other} can be founded.

Remark: Another possible extension of the overlay function can be traced back to the idea of the modified social force (see page 8). As concluded by [64] that higher space requirement will be claimed with the increase of particle speed, in the modelling we may request that, toward a specific particle, the overlay value can be given a certain magnifying factor relating to the former particle's speed. By this means, the target position associated with the overlay value will become unfavourable for the incoming particle, which in turn confirms the increased space requirement at this grid cell position.

Our model requires no further physical quantities retrieved from empirical data, yet it is able to reproduce the interacting behaviour of pedestrians in groups. If the model is equipped with an advanced navigation module and properly calibrated with further information about particles' default speed and its distribution, we are optimistic that more satisfying results can be achieved.

this experiment could be repeated.

3.2 Path-oriented coordinate system

Often pedestrians have a more or less fixed walking direction in their environment. This walking direction may be given explicitly or is the result of local navigation under the environment settings, the latter is especially the case when pedestrians walk along a path, corridor etc. Regular grids, generally speaking, are inadequate for the modelling of paths, since in nontrivial cases the main direction—called *orientation* in the current text—of the path can be often different from the axial directions of the grid.

[19] argued from empirical observations that for pedestrians walking along a path there is a fictional "path force" which draws the pedestrians toward the centre of the path. Thus, in addition to the fictional axis represented by the orientation of the path, a second axis can be defined by the aforementioned path force. This second axis measures the deviation from the centre of the path and is always locally perpendicular to the orientation. In consequence, once we have the orientation of the path and the corresponding deviation, positions can be properly defined in a so-called *path-oriented coordinate system*.

Assuming that the centre line of the path is a plane curve S, the whole area of this path is then in mathematical sense a *tubular neighbourhood* of the curve S. In addition, we request that this curve be regular and have an unchanged sign of curvature.

Remark: On the path associated with a irregular curve, this curve's critical points (where derivative of the curve vanishes or becomes undefined) represent the positions which need to be handled separately. Crossings and junctions are examples of these. If the curvature has changing signs, the centre line of the path can be first decomposed into curve segments with curvatures of an unchanged sign; after discretization, the components of the original curve can be "glued" together to restore the original geometry.

3.2.1 Tubular neighbourhood

We give a formal description of tubular neighbourhood. Let S, the centre of the original path with a certain orientation, be given by a C^1-mapping $S : I \to \mathbb{R}^2$ from an interval $I \subset \mathbb{R}$ into $\mathbb{R}^2$. Let N_S denote the normal vector of the curve S. The deviation from the centre line of the path can be measured by a number $r \in \mathbb{R}$:

$$rN_S.$$

At a local position on the curve $S(t)$ (with $t \in I$), the normal segment can be defined as:

$$N_{S,\delta}(t) = \{\, S(t) + rN_S(t) \mid r \in (-\delta, \delta) \,\},$$

for a given $\delta > 0$.

The tubular neighbourhood $N_{S,\delta}$ is then the union of disjoint normal seg-

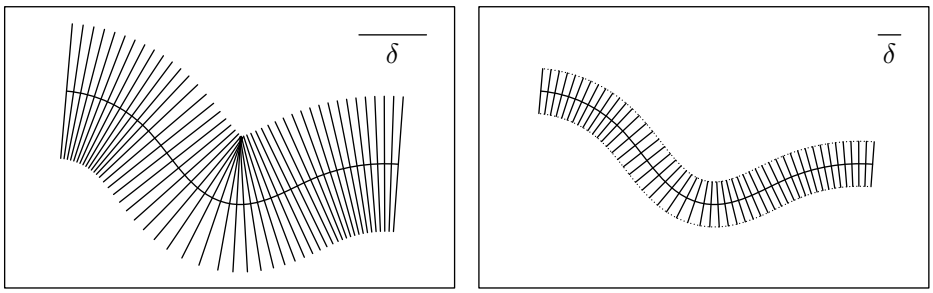

Figure 3.16: Tubular neighbourhood. Left: Normal segments of an arbitrary curve segment with a large δ; the normal segments are not disjoint. Right: Tubular neighbourhood of the same curve segment with a proper δ, F is in this case injective on $S(I) \times (-\delta, \delta)$.

ments of the curve, given by an injective mapping F on $S(I) \times (-\delta, \delta)$,

$$\begin{aligned} &F : S(I) \times (-\delta, \delta) \to N_{S,\delta}(I), \\ &F\left(S(I) \times (-\delta, \delta)\right) = N_{S,\delta}(I) = \bigcup_{t \in I} N_{S,\delta}(t). \end{aligned} \tag{3.20}$$

When δ is no larger than the radius of the local curvature everywhere on S, the injectivity of F can be guaranteed, see Figure 3.16 for a bad and as well as a good example of this. In the sequel, we request a nonzero curvature of S.

3.2.2 Discretization scheme

Obviously the deviation in a path-oriented coordinate system must be no larger than the δ required in (3.20). In other words, the path can be spanned by a maximum deviation $\delta > 0$ in both directions.

Let m be a positive even number and we assume that the path orientation S is known. We can divide the path into m stripes separated by $m+1$ equidistant curves. These equidistant curves will be labelled as S_0, $S_1, \ldots, S_m$, the indices $0, \ldots, m$ start from the inner boundary of the original path. In nontrivial cases (that is, curvature of S being nonzero), the equidistant curves in the inner side have shorter lengths. We divide each equidistant curve S_i $(i = 0, \ldots, m)$ further by arc length into $n + i$ parts. The innermost curve S_0 of the path is then composed of n curve segments whereas the outermost curve S_m, $m + n$ curve segments. These curve segments can be easily approximated by straight line segments. By this means, the original path is represented by a set of triangles, see the left subfigure of Figure 3.17 for an example. This topology can be reorganized to fit in Euclidean coordinate system in a way that any two triangles bounded by two neighbouring equidistant curves sharing a common edge will now form a quadrilateral. These quadrilaterals can be labelled by their x- and y-positions, for $x = 0, \ldots, m-1$ and $y = 0, \ldots, n+x$. Furthermore, in each of these quadrilaterals

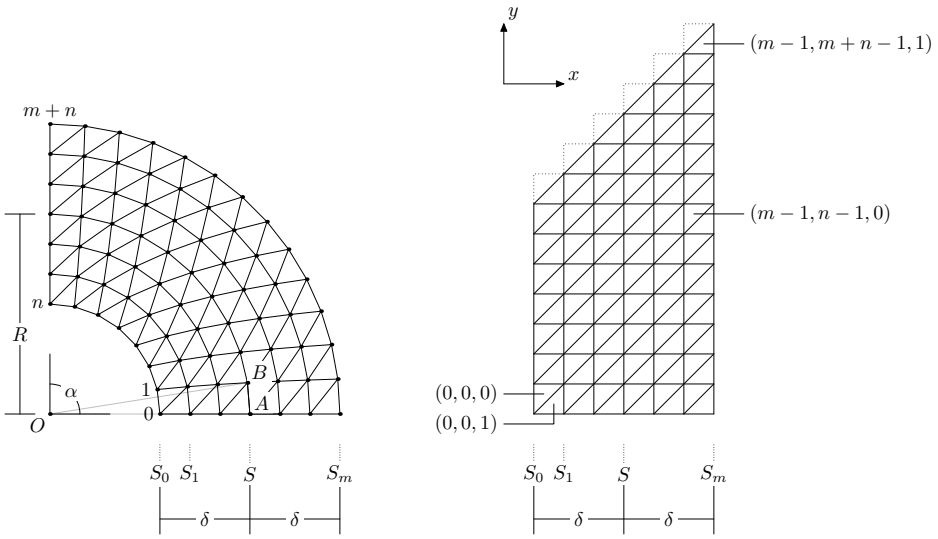

Figure 3.17: Discretization of an arc-shaped area. Left: Discretization in a path-oriented coordinate system with parameters $m = 6$ and $n = 7$. On S, the centre of the path, which is also identical to $S_{\frac{m}{2}}$, the distance between two neighbouring segmentation points is $2R \cdot \sin \frac{\alpha}{2n+m}$. Right: Representation in Euclidean coordinate system.

the triangle in the negative direction of x is assigned with an additional bit 0 and the one in the positive x-direction, an additional bit 1. The original positions, now transformed into triangles in the path-oriented coordinate system, can be properly addressed by 3-tuples, as the right subfigure of Figure 3.17 shows. The reorganization of the curve segments in Euclidean coordinate system serves purely the purpose of an easy labelling of the triangles in the original geometry.

Although m and n are independent parameters in the discretization process, they should be selected in a way that the quadrilaterals composed of neighbouring triangles have a form close to a square whenever possible. This is essential in the discrete modelling of pedestrian dynamics, although their walking directions may or may not be axis-parallel in a regular grid. Now as in a special case, we assume further that S is an arc with a central angle of α and radius of R; in fact, an arbitrary path can always be approximated by arcs of this kind.

Since the centre of the path S will be divided into $n + \frac{m}{2}$ parts, in the ideal case, we request that the line segments on the centre have the same length as the distance between two neighbouring curves. Referring to Figure 3.17, the requirement in the ideal case would become:

$$2R \cdot \sin \frac{\alpha}{2n+m} = \frac{2\delta}{m}, \tag{3.21}$$

The left side of (3.21) gives the length of the line segments on the centre ($|AB|$

in Figure 3.17, which can be derived from R and $\angle AOB$), this in turn gives:

$$n = \frac{1}{2}\left(\frac{\alpha}{\arcsin\frac{\delta}{mR}} - m\right).$$

The length of each of the $n+i$ approximating line segments on S_i ($i = 0, \ldots, m$) reads:

$$2 \cdot \left(R + \left(\tfrac{2i}{m} - 1\right) \cdot \delta\right) \cdot \sin\frac{\alpha}{2n+2i}. \tag{3.22}$$

Respecting the geometric structure to be modelled, α, R and δ are already known, whereas the length $d = \frac{2\delta}{m}$ should resemble the usual step length of the pedestrians in discrete modelling, which usually takes the value of $l_c = 0.4\,\mathrm{m}$ (see § 3.1.2 and § 3.1.6.1); thus m can be decided in this way. We then need to check the difference of (3.22) to the specific length $\frac{2\delta}{m}$ in (3.21) to see whether it lies within a pre-defined tolerance range; if not, the parameter choice of α should be adjusted. If locally the curvature of the path orientation (that is, $\frac{1}{R}$, respecting S) changes too sharply, the path area can be divided into smaller pieces before being further approximated.

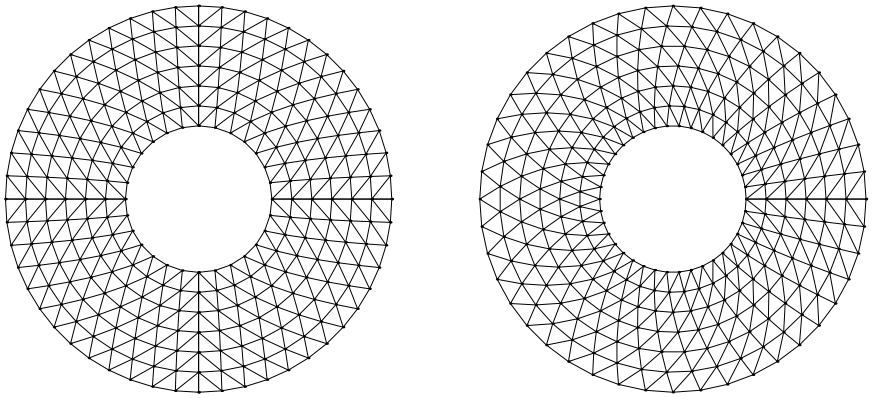

Figure 3.18: Triangulation of a ring-shaped area in path-oriented coordinate system. Left: The triangulation is further composed of four pieces of triangulation of a $\frac{\pi}{2}$-arc. Right: The triangulation is carried out on the circle area defined by a 2π-arc; the length difference among the line segments is too large.

Remark: We recall that in common discrete models based on regular grids, the pedestrian enjoys an exclusive personal space of l_c^2 ($l_c = 0.4\,\mathrm{m}$); in our case, however, in each quadrilateral composed of two small triangles two pedestrians will be allowed to be present. Hence, an additional factor $\sqrt{2}$ should be considered and we set $d = 0.566\,\mathrm{m}$.

Figure 3.18 shows the decomposition of a ring-shaped area to obtain a triangulation. The figure contains both a good and a bad example. In general, by our discretization scheme, an arbitrary path can be decomposed and approx-

imated into a series of arc-shaped sectors, which can be further represented by structured triangular grids.

3.2.3 Position transition

We now consider position transition in the path-oriented coordinate system. Basically, there are two types of elementary operations. The first kind of position transitions take place in the direction of the path orientation and the second, in the direction of its deviation.

3.2.3.1 Primary direction

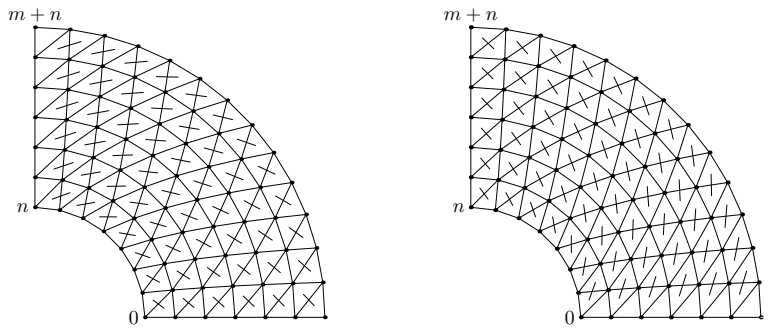

Figure 3.19: Position transition in the primary direction. Left: Forward transition from start position with an additional bit 1 or backward transition from start position with an additional bit 0. Right: Forward transition from start position with an additional bit 0 or backward transition from start position with an additional bit 1.

In the primary direction, position transition takes place in the same stripe bounded by two neighbouring equidistant curves S_i and S_{i+1} $(i = 0, \ldots, m-1)$. Obviously, a forward position transition can be defined as:

$$\mathrm{Fwd}(x, y, b) = \begin{cases} (x, y+1, 1), & \text{if } b = 0, \\ (x, y, \quad 0), & \text{if } b = 1. \end{cases}$$

Since the position transition takes place in the same stripe of the path, the result of a Fwd-operation is a change in the y-component and the additional bit of the position, Fwd is always defined, see Figure 3.19 for an illustration. Backward position transition can be formulated in an analogous way.

Positions $(x, 0, 1)$ and $(x, n+x, 1)$, for $x = 0, \ldots, m-1$, represent the border in the primary direction and will be involved when two path areas are connected.

3.2.3.2 Secondary direction

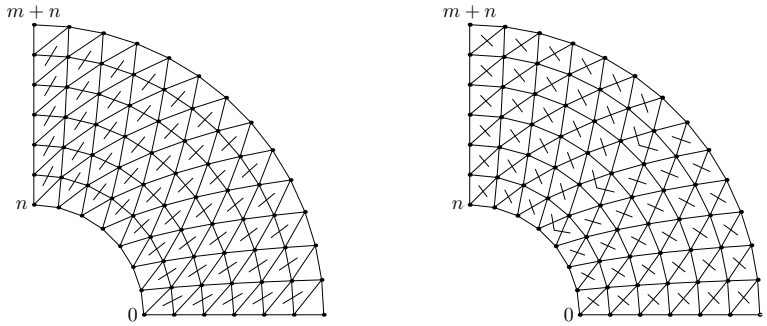

Figure 3.20: Position transition in the secondary direction. Left: Position transition involves a neighbouring stripe. Right: Position transition in the same stripe; in this example, respecting positions $(0, 3, 0)$, $(2, 4, 0)$ and $(4, 5, 0)$ there would exist two possibilities for the operation Dev^+; for $(1, 4, 1)$, $(3, 5, 1)$ and $(5, 6, 1)$ the operation Dev^- is undefined.

Under the influence of the so-called path force, pedestrians move in the secondary direction of the path-oriented coordinate system, which is always locally perpendicular to the path orientation. Position transition in the secondary direction is slightly complicated, since the new position may or may not be in the same stripe as the start position. The direction from the inner side to the outer side of the path will be referred to as the positive direction of the deviation. The direction of the deviation will be addressed by the superscript $^+$ or $^-$ explicitly. A positive deviation from a position with an additional bit 1 or a negative deviation from a position with an additional bit 0 can be formulated as:

$$\begin{aligned} \mathrm{Dev}^+(x, y, 1) &= (x + 1, y, 0), \\ \mathrm{Dev}^-(x, y, 0) &= (x - 1, y, 1), \end{aligned} \tag{3.23}$$

see the left subfigure of Figure 3.20. (3.23) involves the position change into a neighbouring stripe.

Positive deviation from a position with an additional bit 0 and negative deviation from a position with an additional bit 1 are illustrated in the right subfigure of Figure 3.20. In this case, the position transition takes place in the same stripe:

$$\mathrm{Dev}^+(x, y, 0) = \begin{cases} (x, y, \quad 1), & \text{if } y < \frac{n+x-1}{2}, \\ (x, y + 1, 1), & \text{if } y \geq \frac{n+x}{2}, \end{cases} \tag{3.24a}$$

$$\mathrm{Dev}^-(x, y, 1) = \begin{cases} (x, y, \quad 0), & \text{if } y \leq \frac{n+x-1}{2}, \\ (x, y - 1, 0), & \text{if } y \geq \frac{n+x+1}{2}. \end{cases} \tag{3.24b}$$

We notice that (3.24a) and (3.24b) are not defined for every position in y. This is the case for the centre of each stripe. If $n+x$ is an odd number, (3.24a) is undefined for $y = \frac{n+x-1}{2}$, in fact, $\mathrm{Dev}^-(x, \frac{n+x-1}{2}, 0)$ can be either $(x, \frac{n+x-1}{2}, 1)$ or $(x, \frac{n+x+1}{2}, 1)$. If $n+x$ is an even number, (3.24b) is left intentionally undefined for $y = \frac{n+x}{2}$, since $(x, \frac{n+x}{2}, 1)$ has no neighbouring position in the negative direction of the deviation in the path-oriented coordinate system with which a common edge is shared, see the right subfigure of Figure 3.20.

3.2.4 A test application

In this section, we present a test application to demonstrate our idea. We consider the particle flow on a closed ring-shaped area (see Figure 3.18) with periodic boundary in the y-direction[13]. We assume that parameters α, R and δ, and consequently m and n, are known. The particle speed v in the primary direction is governed by (3.19).

3.2.4.1 Adaptation of continuous density

First, the continuous density on discrete positions (3.16) needs to be adapted on our path-oriented coordinate system as a triangular grid. If we can transform an arbitrary position (x, y, b) into two-dimensional coordinates (x', y'), we will be able to reinvoke (3.16):

$$\hat{\rho}_I(x, y, b) := \hat{\rho}_I(x', y').$$

Given a position (x, y, b), our suggestion is to eliminate the additional bit b and replace x and y by new positions in the primary and second directions x' and y'. In the x-direction, we request:

$$x' = x + \beta,$$

where

$$\beta = \begin{cases} \frac{1}{4}, & \text{if } b = 0, \\ \frac{3}{4}, & \text{if } b = 1, \end{cases} \tag{3.25}$$

is a parameter to address the position shift in the quadrilaterals of the discretization. By this means, an equal distance in the x-direction can be maintained for neighbouring positions with a same y-coordinate.

[13] See also page 60 for an explanation of periodic boundary in the case of square grid.

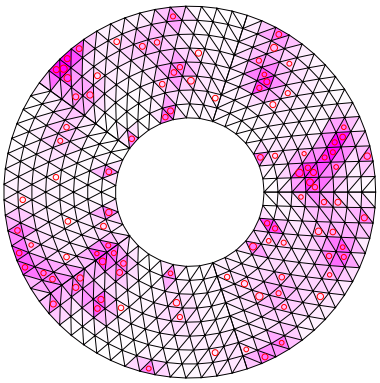

Figure 3.21: Continuous density on a ring-shaped area in a path-oriented coordinate system, particles are initialized with random positions.

In the y-direction, let us consider the intersection point with the original path orientation S and the straight line connecting the origin of the arc sector and the current position (x, y, b). In such a way, y' can be understood as the projection of y-component of the position on the original path orientation S. We suggest a replacement of y by:

$$y' = \frac{n + \frac{m}{2}}{n + x + \frac{1}{2}} \cdot (y + 1 - \beta),$$

with β defined in (3.25).[14]

Figure 3.21 gives an example of the continuous density on path-oriented coordinate systems, again we notice that a single isolated particle does not necessarily induce a high density (cf. Figures 3.9 and 3.10), whereas a particle at the inner or outer boundary induces a higher density value, since the corresponding d_i-value (distance to the nearest obstacle) is low in such a case.

3.2.4.2 Local navigation by path force

Apart from the position transition in the primary direction which is determined by the particle's default speed, navigation in the secondary direction must be considered as well.

Let Δx denote the position transition in the secondary direction in a simulation cycle. In discrete form, the path force proposed by [19] can be written as:

$$\Delta x = \left(\frac{m - 1}{2} - x \right) \cdot a, \tag{3.26}$$

where a is a control parameter. In (3.26), Δx represents a secondary speed component which always points to the centre of the path. Unfortunately, calibration of a is not provided in the original work of [19].

On the other hand, we observe that the inner boundary has a shorter geometric length, it will not be unreasonable to assume that pedestrians choose to walk shorter distances when the situation allows. Generally, the effect of the

[14] $n + \frac{m}{2}$ is the number of the line segments on the centre, $n + x + \frac{1}{2}$ gives an estimate of the discrete length of the stripe containing the current position; $y + 1 - \beta$ offers a correction of the y-component with consideration of the additional bit b, cf. Figure 3.17.

so-called path force can be written as:

$$\Delta x^{(\chi)} = (\chi - x) \cdot a. \tag{3.27}$$

$\chi = \frac{m-1}{2}$ presents the special case of (3.26); whereas $\chi = 0$ refers to the case of the preferred way choice of the inner side of the boundary.

In our model, Δx will be considered as the maximum *possible* position transition in the secondary direction. In each simulation cycle, every particle searches for a triangular cell within the range of Δx at its default speed with an appropriate density overlay value, as in the model on rectangular grids in § 3.1.6.2. In other words, particles' range of perception, within which choices on the tactical level are to be made, is decided by the magnitude of the path force. On the operational level, for the execution of the elementary steps in both primary and secondary directions we refer to § 3.1.3.2, as in the case of rectangular grids.

We notice that, in the case of $\chi = 0$, Δx by (3.27) will be always be either negative or zero. To avoid unrealistic jamming, we extend the particles' perception range by one quadrilateral on the other side of the direction pointed by Δx. In case $\Delta x = 0$, the particles' perception range will be composed by one quadrilateral on both sides in the secondary direction.

For simplicity's sake, and owing to the absence of empirical data for model validation, density overlay without threshold values (3.17) will be applied.

Remark: Apart from a, the magnitude of m has an effect on the strength of the path force as well. This is to say that a should be configured with consideration of m in the individual models.

3.2.4.3 Simulation results and discussion

We show test results of the simulation. In Figures 3.22 and 3.23 fundamental diagrams of density and flow are given. Density is measured on the lane basis (see § 2.3.5.3.1), the only difference is that here we have "stripes" (areas between two neighbouring equidistant curves) instead of "lanes" in the system geometry. Since the flow of the particles is closed within the ring-shaped area, it is reasonable to set a comparatively high population of the particles, as long as this does not unavoidably lead to a system-wide jam of the flow in the simulation. In the examples, the triangular cells are given a probability of 40% to be populated by particles during the system initialization.

Similar to § 2.3.5.3, relative linear scale is applied in the figures. The horizontal label "1" refers to the maximum attainable density of the particles, which is $2.26\,\mathrm{m} \cdot \mathrm{s}^{-1}$ (see § 3.1.6.1); the vertical "1" represents the product of density and the average default speed of the particles. Since it is assumed that each pedestrian has an exclusive space of $0.4\,\mathrm{m} \cdot 0.4\,\mathrm{m}$, if we further assume that

the representing particles pass one by one in the flow direction—to be more specific, in a passage of width 0.4 m—, the maximum theoretical flow volume would be

$$\frac{1}{0.4\,\mathrm{m}} \cdot 1.33\,\mathrm{m} \cdot \mathrm{s}^{-1} = 3.325 \text{ (particles) } \mathrm{s}^{-1}.$$

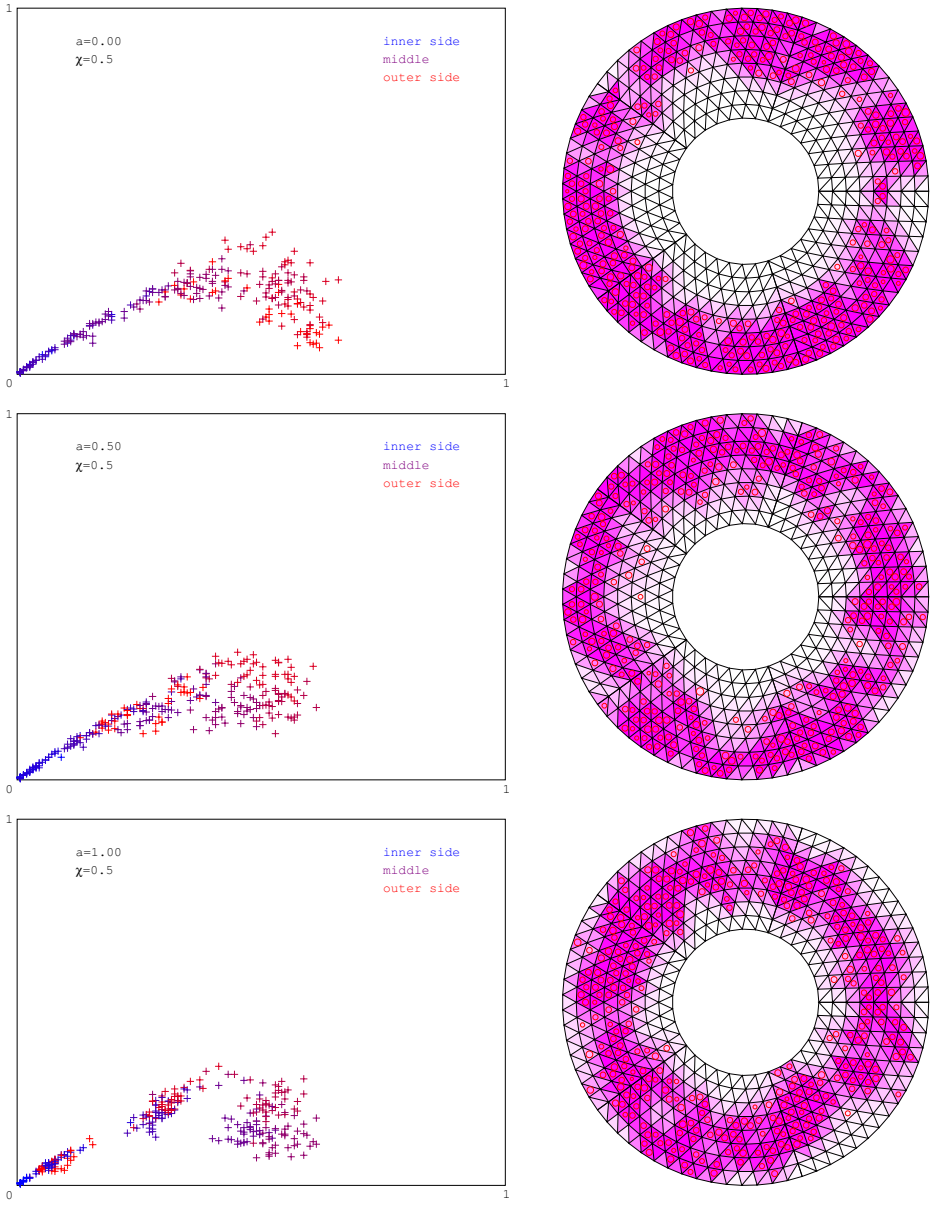

Figure 3.22: Fundamental diagrams and dispersing of the particles in the case of $\chi = \frac{m-1}{2}$.

Figure 3.22 shows the cases of zero path force, half and full path force toward the centre of the path. With zero path force, fundamental diagram of the traditional shape can be generated. The transition from upstream to downstream

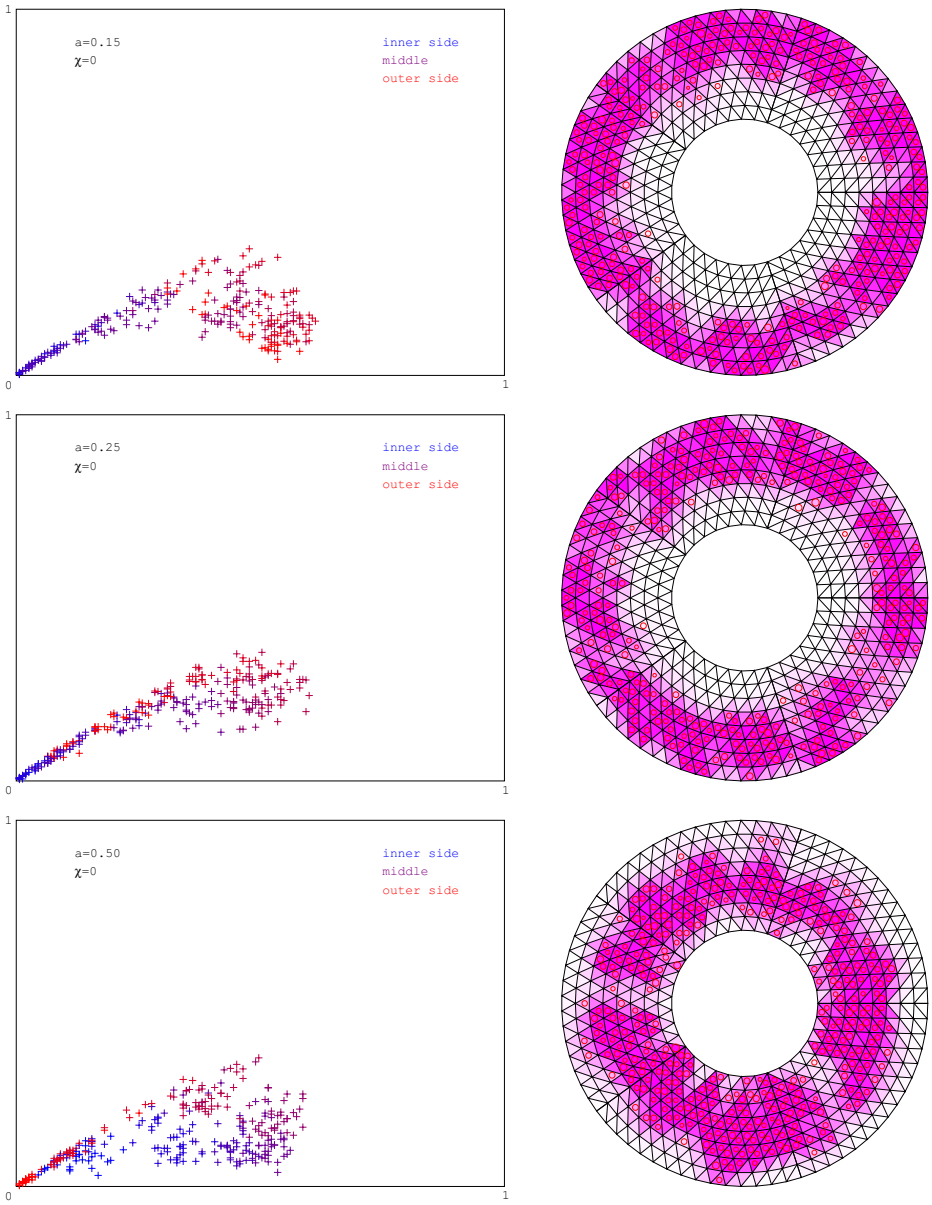

Figure 3.23: Fundamental diagrams and dispersing of the particles in the case of $\chi = 0$.

is seen from the inner side of the path to the outer side. With nonzero path force we observe a slight but gradual jamming in the centre of the path, this is to be understood that the constant $a = 0.5$ or 1 is set too high in the simulation.

Technical note. With zero path force, particles tend to flow to the outer side (and middle) of the path, this is because that when initialized with a random particle population, more particles have a start position in outer stripes, overall this would lead to a higher

continuous density in the outer side. Particles in the simulation are navigated by other members of their own group, therefore they shift generally to the outer side of the path.

Figure 3.23 shows the case of path force pointing to the inner side with $a = 0.15, 0.25, 0.5$. It appears that for $a \leq 0.25$ that path force is within a reasonable range. With path force pointing both to the centre and to the inner side, a peak flow volume of roughly one particle per second has been observed; in the latter case this peak is reached at a slightly lower density (at approximately 40% of the maximum possible density).

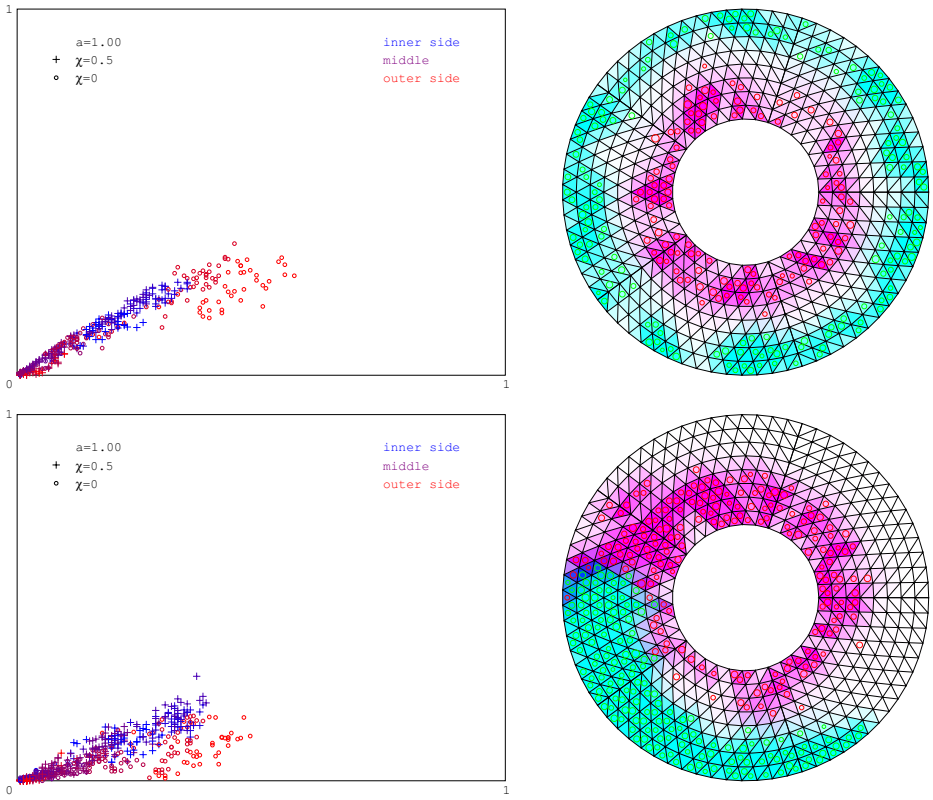

Figure 3.24: Fundamental diagrams and dispersing of the particles of two groups. First row: Particle of both groups are populated with a probability of 15% during the initialization. Second row: Particles of both groups are populated with a probability of 20%. Sample points ("+" and "∘") of the two groups are shown separately.

In Figure 3.24 we show the simulation results with two groups of opposite moving directions. In a closed system, even a relatively low density would lead to complete congestion caused by the conflicting particles of the two groups. This is because with the system geometry being a ring-shaped area, the path will not be wide enough even for a moderate overall population density. In such a case,

the simple navigation strategy by avoidance of obstacles and following one's own group, as expressed by the overlay in (3.17), cannot guarantee a free flow. In our test, maximum path forces toward the inner side and outer side are defined for the two particle groups, this resembles the situation in which pedestrians of the two groups are given additional instructions to prefer the utilization of a specific part of the path. However, even with this, a slightly higher density (initial population probability at 20% for both groups) still leads to complete congestion.

The geometry of the tests in this section is composed of five identical arcs ($\alpha = \frac{2\pi}{5}$), further tests show no significant change in the statistical aspects by varying the angle α, as long as the geometry is properly constructed (see page 73).

In this chapter we presented some grid-based methods for the modelling of pedestrian dynamics. Originally in the microscopic category, various cellular automaton models had been proposed for pedestrian dynamics. Rather complex rules had been introduced to describe the interaction among pedestrians. In our grid-based approach, applicable on similar geometries, we proposed a local balancing mechanism and a navigation method by continuous density to describe the interaction among pedestrians in groups. These have been operated on both regular geometric structures represented in square grids and on the so-called path-oriented coordinate systems. Although we may not encounter the latter very often in real-world situations, the idea behind it is to offer a possibility of describing irregular geometries by discretization and recomposition (or integration), when necessary, with other regular structures. Overall, the grid-based methods presented here can be implemented for general purposes.

Chapter 4

Trajectory Extraction

Mathematical models devised for practical uses need to be validated or at least calibrated with empirical data. In the context of the current work, counting of pedestrians offers a simple and straightforward way to derive information about flow and density in a certain test environment, which can be further deployed in the study of fundamental diagrams. Measurements to this effect have a macroscopic flavour. In microscopic analysis, especially in the study of pedestrian walking behaviour, input of visual data will become indispensable [2]. Concerning an object, trajectory is the record of state evolution concerning its spatial position over a certain time span. With the trajectories of all the pedestrians engaged, the system dynamics can be reconstructed without information loss. In particular, counting for the purpose of macroscopic evaluation of the system as stated above becomes trivial.

Visual data are often recorded at a rate of thirty frames per second. Even of a short time length, a sequence would demand tremendous human effort for the calculation of its trajectory information. From a practical perspective, automated methods are indispensable for this task, these have indeed become an independent research field with a very broad spectrum (see [74, 33] for an overview on the topic). The general object tracking problem is composed of object representation, feature extraction, object detection and recognition, and object tracking. Specifically, object tracking as the last step of these can be point-based (with the Kalman filter [8] as a typical ansatz), kernel-based or silhouette-based. Kernel-based tracking is applied on the characteristics of the motion, in terms of consecutive frames, of a so-called template representing an object. Silhouette-based (also called contour-based) tracking is suited for complex geometric objects when templates with a simple geometry reach their limit. Computation of templates and complex contours can be carried out on the basis of optical flow [29, 43] or partial differential equations [5, 44]. Sometimes occlusion of objects must be given consideration additionally. Since tracking is out of the scope of the current text, we will focus on two unconventional approaches for the task of trajectory generation.

4.1 A photogrammetric approach

In [54], we combined analytical techniques borrowed from photogrammetry with the renowned Lucas-Kanade method [43] for the tracking of pedestrians. The Lucas-Kanade method is based on the computation of optical flow (see discussion in the last passage): the method establishes equations of motion respecting the image content in the neighbourhood of a given point within a series of input frames, assuming a smooth transition of the corresponding image content along the time axis. Traditionally, many efforts would have been expected in the calibration and pre-processing of the input visual data in order that these may be further processed by tracking algorithms. The method of [54] has a flavour of exerting "brute force" on datasets with which little or no calibration has been undertaken: it attempts to retrieve the original three-dimensional coordinates via the two-dimensional information available from the data recordings.

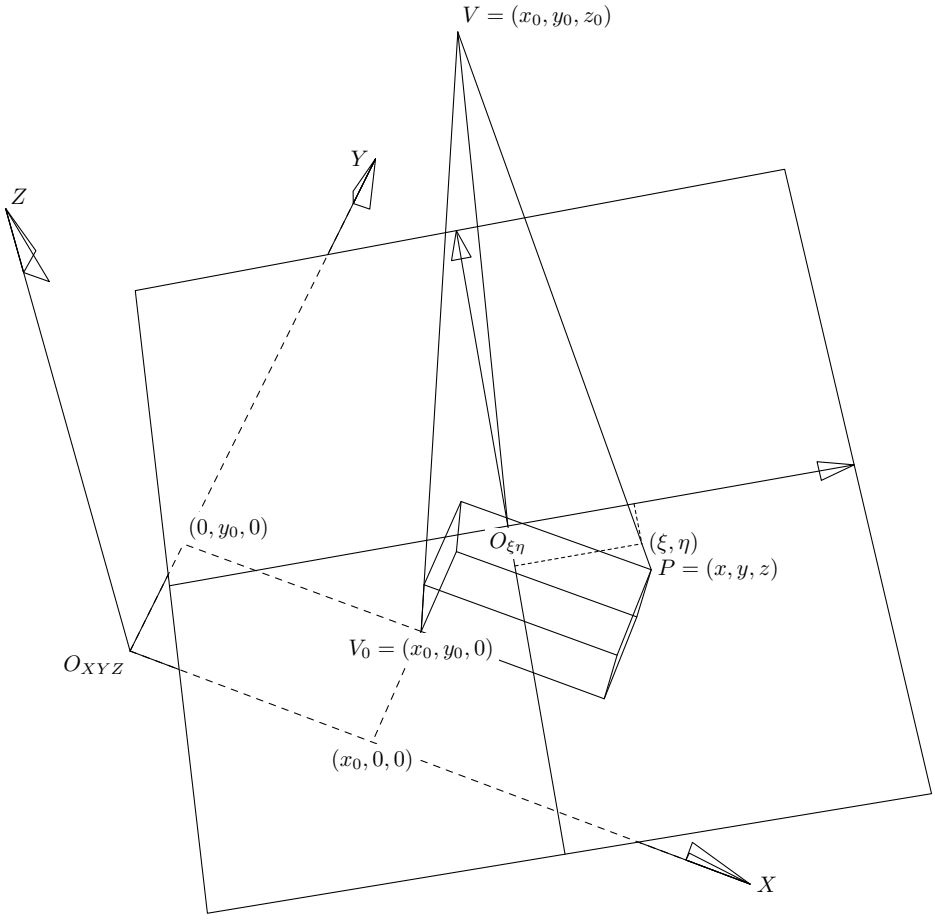

Figure 4.1: Perspective projection of a three-dimensional point P onto the ξ-η-plane. This illustration is itself in perspective projection. Note that V_0V is parallel to $O_{XYZ}Z$ and $O_{\xi\eta}V$ is orthogonal to the ξ-η-plane. V is the view point of the projection.

To be specific, we need to reverse-engineer the process of digital photography from a mathematical perspective, that is, we need to know how an arbitrary point $P = (x, y, z)$ in three dimensions is projected onto an arbitrary plane with coordinates of ξ and η. In the common literature we found only schematic drawings on this topic, we therefore venture to present our own mathematical illustration in Figure 4.1.

Let $V = (x_0, y_0, z_0)$ be the view point of the projection. Let the distance from V to the ξ-η-projection plane be d. Obviously, we request $d \neq 0$. We define the origin of the projection plane $O_{\xi\eta}$ to be the projection of the view point V on the ξ-η-plane. By collinearity, the following relationship concerning the two-dimensional coordinates (ξ, η) and the three-dimensional coordinates (x, y, z) can be established:

$$\begin{aligned} \xi &= -d \cdot \frac{r_{11}(x - x_0) + r_{21}(y - y_0) + r_{31}(z - z_0)}{r_{13}(x - x_0) + r_{23}(y - y_0) + r_{33}(z - z_0)}\,, \\ \eta &= -d \cdot \frac{r_{12}(x - x_0) + r_{22}(y - y_0) + r_{32}(z - z_0)}{r_{13}(x - x_0) + r_{23}(y - y_0) + r_{33}(z - z_0)}\,. \end{aligned} \tag{4.1}$$

In (4.1), $r_{11}, \ldots, r_{33}$ are the coefficients of the corresponding rotation matrix by which the original three-dimensional plane $XO_{XYZ}Y$ is transformed into the ξ-η-projection plane and the vector $O_{XYZ}Z$ into a new vector $O_{\xi\eta}V$. The rotation matrix

$$R = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}$$

can be written expressly as:

$$R = \begin{pmatrix} \cos\phi\cos\kappa & -\cos\phi\sin\kappa & \sin\phi \\ \cos\omega\sin\kappa + \sin\omega\sin\phi\cos\kappa & \cos\omega\cos\kappa - \sin\omega\sin\phi\sin\kappa & -\sin\omega\cos\phi \\ \sin\omega\sin\kappa - \cos\omega\sin\phi\cos\kappa & \sin\omega\cos\kappa + \cos\omega\sin\phi\sin\kappa & \cos\omega\cos\phi \end{pmatrix},$$

whereas the rotation angles ω (primary), ϕ (secondary) and κ (tertiary) are associated with the X-, Y- and Z-axes respectively.[1]

To reconstruct the original three-dimensional scene, concerning every unknown point $P = (x, y, z)$, its spatial translation $(x - x_0, y - y_0, z - z_0)$ would be sufficient. Thus, (4.1) contains two equations with at most ten unknown parameters ($r_{11}, \ldots, r_{33}$ and d). We recall the fact that the nine parameters $r_{11}, \ldots, r_{33}$ have a freedom degree of three, since they are derived from the rotation angles ω, ϕ and κ. Given adequate reference points in both coordinate systems, (4.1) can

[1] Viewing from the positive end of the corresponding axis (that is, with the axis pointing to the observer), the rotation angle runs counter-clockwise.

be solved via additional approximation procedures, for example, by the method of least squares. Thus our problem can be considered as mathematically solved. In [55], this photogrammetric method has been the basis of multi-view extraction of pedestrian trajectories.

Since no object detection component was introduced in the implementation of [54], the start positions (points) for the Lucas-Kanade method have to be manually defined in the tracking; overall, this poses as a semi-automatic solution for the problem of trajectory extraction.

4.2 A Time-Of-Flight approach

The second approach is based on the modulation-based Time-Of-Flight (TOF) technology. TOF is a contactless ranging (distance measurement) technology. We will give here only the very basic introductory notes, since ranging is not the topic of the current research work.

Traditionally, radar[2] has been widely deployed to detect moving objects. A radar system emits radio waves (also called *signal*s) to be reflected and scattered by the objects to detect. A receiver of the system measures the intensity of the reflected signals, with consideration of additional parameters, for example, the reflection angle and the size of the receiver surface, the range and altitude of the object can be computed and further physical quantities can be inferred. This can be considered as a direct ranging method. In comparison, triangulation and interferometry have been two other traditional but indirect ranging methods. More or less, triangulation is a mathematical approach, whereas interferometry, a physical method: with triangulation, distance measurement of a point is carried out by measuring angles to it from auxiliary points whose positions are already known, the distance to measure will be found in the solution of a geometry problem; with interferometry, waves (mostly electromagnetic) are superimposed to render different light patterns called interference fringes, by means of which precise wave length information can be determined, distance can be thus calculated from the latter. Interferometry can be applied in distance measurement from microscopic to astronomical scales. We refer to [6] for an overview of techniques applied in range imaging sensors in which further methods are explained.

4.2.1 A brief introduction to TOF

Time-Of-Flight imaging is a rather direct approach, it is based on the measurement of light travel duration.

[2] Acronym for "RAdio Detection And Ranging".

Pulse-based TOF technology measures the light travel duration (from which the term "Time-Of-Flight" is coined) directly [50, 51]. In such a system, the reflected light from an illuminated scene is captured by single-photon avalanche diode (SPAD) detectors working on a scale of picoseconds. SPAD detectors allow the use of light sources with low intensity which has a substantial cost benefit respecting the light emission device and at the same time, it offers operation safety for humans. With this method, however, extremely high precision in time measurement becomes a prerequisite.

The other category of TOF systems is modulation-based (sometimes called "continuous-wave modulated") [63, 40, 53, 36, 21]. This method measures the phase shift between an amplitude-modulated light wave (in most cases emitted by a solid-state laser[3]) and its reflection—instead of the direct flight duration—to calculate the distance. Within a certain distance range, this phase shift $\Delta\phi$ is proportional to the flight duration τ:

$$\Delta\phi = 2\pi f\tau, \tag{4.2}$$

and hence, proportional to the distance d to measure:

$$\Delta\phi = \frac{4\pi f}{c} \cdot d, \tag{4.3}$$

where f is the modulation frequency and c the speed of light. In (4.2), $\Delta\phi$ is unambiguous within the range $d_{\max} = \frac{c}{2f}$, which guarantees the validity of (4.3). A typical modulation frequency $f = 20\,\mathrm{MHz}$ would lead to an unambiguous distance range of $d_{\max} = 7.5\,\mathrm{m}$. To extend this range, f can be lowered. The reduced accuracy in measurement owing to this may be compensated by higher illumination.

Accompanied by the development of the CMOS (complimentary metal-oxide-semiconductor) technology, the "on-chip" implementation of this idea was introduced [67]. In such a system, a CMOS pixel array will be deployed to capture photonic energy and convert it into electric current.

According to [13] and the references contained therein, for a modulated light, four samples per period would be enough to reconstruct its amplitude and phase. In four equal sample intervals, the photonic charge, which is proportional to the intensity of the incoming light, will be integrated respectively within one modulation period. This process can be repeated over many modulation periods to detect low intensity signals.

Let $I_0, \ldots, I_3$ denote the four intensity values sampled at $\phi_0, \ldots, \phi_3$ ($\phi_j =$

[3] Acronym for "Light Amplification by Stimulated Emission of Radiation".

$\phi_0 + \frac{\pi j}{2}$, $j = 0, 1, 2, 3$) respectively. For the actual phase shift $\Delta\phi$, we have:

$$\Delta\phi = \arctan\left(\frac{I_3 - I_1}{I_2 - I_0}\right), \tag{4.4}$$

whereas the amplitude would be:

$$I = \frac{\sqrt{(I_3 - I_1)^2 + (I_2 - I_0)^2}}{2}.$$

By (4.3) and (4.4), the distance d is known. A closer look of (4.4) reveals that, instead of the four sampled values $I_0, \ldots, I_3$, only the differential terms $I_2 - I_0$ and $I_3 - I_1$ will be needed for the computation. [17] pointed out that the standard deviation of the phase-range measurement is directly reciprocal to the intensity. This leads to the conclusion that measurements derived from a too low intensity can be safely discarded.

In the process of measurement, the CMOS pixel array generates a distance map, since at every pixel position (m, n) a radial distance value d is now computed. The transformation of this information into the Cartesian coordinates (x, y, z) of the sensor system reads:

$$\begin{aligned} x &= \frac{md}{\sqrt{\alpha^2 + m^2 + n^2}}, \\ y &= \frac{nd}{\sqrt{\alpha^2 + m^2 + n^2}}, \\ z &= \frac{\alpha d}{\sqrt{\alpha^2 + m^2 + n^2}}, \end{aligned} \tag{4.5}$$

where m and n denote the respecting one-dimensional distances from the chip centre measured in pixel units. The origin of this new transformed coordinate system overlaps with the centre of the pixel array, and its z-axis is parallel to the optical axis of the emitter of the TOF system. $\alpha = \frac{l_f}{l_p}$ denotes the ratio of the (imaginary) focal length of the emitter l_f and the physical edge length of the pixel—that is, the pixel size in one dimension l_p—on the chip.

In the area close to the centre of the chip, (4.5) can be approximated as

$$x \doteq \frac{md}{\alpha}, \quad y \doteq \frac{nd}{\alpha}, \quad z \doteq d.$$

Away from the centre, the above approximation may result in unfavourable optical distortion, if no additional correction is provided.

4.2.2 System installation

We choose to install the TOF sensor to be perpendicular to the floor of the scene. This has the advantage that the sensor coordinate system can be easily transformed into a world coordinate system to which the observers are more

accustomed.

Let the distance from the sensor to the floor be h_0 ($h_0 > 0$). Then the floor in the scene becomes the plane $z = h_0$ in the Cartesian coordinate system of the sensor described in (4.5). The new transformation can be easily given by the following mapping:

$$x \mapsto x, \quad y \mapsto y, \quad z \mapsto h = h_0 - z. \tag{4.6}$$

Now in the world Cartesian coordinate system, the floor is the represented by the plane $h = 0$, and naturally, the position projection of the sensor on the floor is the origin $(0, 0, 0)$, see Figure 4.2.

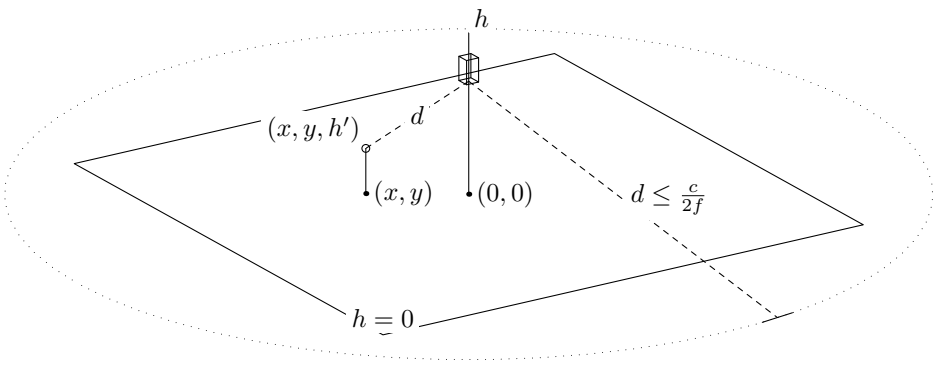

Figure 4.2: The installation of a TOF sensor. By (4.5) the radial distance d will be projected onto the optical axis of the sensor (which is the reverse of the h-axis in the scene). For simplicity's sake, this value will further, by (4.6), be transformed into a height value h in the coordinate system of the scene. The rectangle on the plane $h = 0$ is the area covered by the sensor, or to be more specific, captured by the CMOS pixel array of the TOF sensor.

Depending on the CMOS technology applied in the TOF sensor, the area covered by the sensor may vary but is always a rectangle (as the term "array" suggests). In our context, it suffices to request this area to be convex in geometric sense. Since in the sequel no further condition concerning the geometric shape of the observation area will be imposed, we denote here, without loss of generality, this area of observation by a rectangular grid $\Omega = \{0, \ldots, n_x - 1\} \times \{0, \ldots, n_y - 1\}$, and by n_x and n_y we denote the number of samples in the x- and y-dimension respectively. Every grid position in Ω is associated with a pre-processed height value $H_{x,y}$, ($x = 0, \ldots, n_x - 1$, $y = 0, \ldots, n_y - 1$). Referring to Figure 4.2, the measurement point labelled (x, y, h') is associated with a height value $H_{x,y} = h'$. Indeed, H can be considered as a height function respecting the discrete positions in the observation area covered by the TOF sensor. The boundary of Ω can be written as $\partial\Omega$.

The matrix of the height values

$$\Big(H_{xy}\Big)_{\substack{x=0,\dots,n_x-1\\ y=0,\dots,n_y-1}} \tag{4.7}$$

will be called a raw *image frame* (or simply *frame*) recorded by the sensor.

4.2.3 Preparation in a formal aspect

Not considering measurement error or inaccuracy, objects detected by the TOF sensor always exhibit positive height values in the world coordinate system. In other words, these objects may be considered as being "placed" on the floor plane $h = 0$. The outer[4] surface of an object is in many cases geometrically convex. If the measurement could be undertaken in a continuous manner, the height function H defined on the observation area (written as Ω in discrete form, see the previous passage) would become a continuous function. We observe the following

Theorem 4.1 (Continuous version)**.** *The height function H of a strictly convex object has only one local maximum.*

Proof. Recalling that the observation area, on which H is defined, is convex, we let, without loss of generality, the height function $H : D \to \mathbb{R}$ be defined on a convex set $D \subset \mathbb{R}^2$ and have two different local maxima $H(p)$ and $H(q)$ at $p, q \in D$ $(p \neq q)$. Again without loss of generality, we assume $H(p) \leq H(q)$.

By definition of strict convexity, there exists an $\varepsilon \in (0, 1]$ for the local maximum $H(p)$ such that $H(p') < H(p)$ for all $p' = (1-t)p+t\cdot q$, with $t \in (0, \varepsilon)$. In other words, for any t so chosen, we have $H\left((1-t)p + t \cdot q\right) = H(p') < H(p)$. On the other hand, recalling $H(p) \leq H(q)$, we have, by linearity, for all $t \in (0, 1)$: $H(p) < (1-t)H(p) + t \cdot H(q)$. Combining these two statements, we have $H\left((1-t)p + t \cdot q\right) < (1-t)H(p) + t \cdot H(q)$, which contradicts the convexity of the object. The contraposition of the original claim is herewith proven. □

We recall that the strict convexity of the object is defined on the convex domain of definition D:

$$\forall\, p, q \in D, p \neq q :\ \forall\, t \in (0,1) : H\left((1-t)p + t \cdot q\right) > (1-t)H(p) + t \cdot H(q). \tag{4.8}$$

In real-world scenarios, however, the domain D is a discrete set composed of finitely many sample points $p_0, \dots, p_{n-1} \in \mathbb{R}^2$. In correspondence with the measurement of the object, the height function H is given on these positions discretely. We may divide the original domain D into geometrically convex[5]

[4] We define this to be in the positive h-direction.

[5] This can be achieved by—but not limited to—a Voronoi decomposition. As a result, the Voronoi polygons are always convex by construction.

subdomains $D_0, \ldots, D_{n-1}$ associated with $p_0, \ldots, p_{n-1}$. Let $N = \{0, \ldots n-1\}$ be the index set. Since for every $i \in N$, p_i is the only position where measurement takes place, we can simply define the height function H on the subdomain D_i to be

$$H(D_i) := H(p_i), \tag{4.9}$$

for all $i \in N$. Naturally, for any arbitrary point $q_i \in D_i$, we may define:

$$H(q_i) := H(D_i), \tag{4.10}$$

Intuitively, a position p can be considered as a local maximum, if its height value exceeds all those in the surrounding subdomains. If, in the discrete case, we adapt the definition of strict convexity (4.8) into

$$\begin{aligned} &\forall\, a, b, c \in N, a \neq b, c \neq a, c \neq b : \\ &\quad \exists\, q_a \in D_a, q_b \in D_b :\ \exists\, t \in (0,1) : \\ &\qquad (1-t)q_a + t \cdot q_b = p_c \Longrightarrow \\ &\qquad\quad H(D_c) > (1-t)H(D_a) + t \cdot H(D_b), \end{aligned} \tag{4.11}$$

we can reformulate Theorem 4.1 in the following form:

Theorem 4.2 (Discrete version)**.** *A height function H defined on finitely many sample points, satisfying* (4.11)*, has only one local maximum.*

Proof. Without loss of generality, let p_a and p_b be the sample points associated with subdomains D_a and D_b respectively. Respecting a third subdomain D_c, we consider two points $q_a \in D_a$ and $q_b \in D_b$ which are collinear with the sample point p_c of D_c. By (4.9) and (4.10) we have $H(q_a) = H(D_a) = H(p_a)$ and $H(q_b) = H(D_b) = H(p_b)$, the rest of the proof is trivial. □

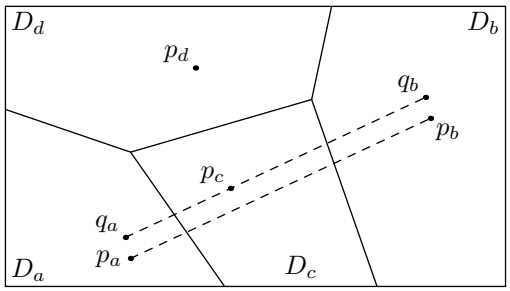

Figure 4.3: Convexity in subdomains. Four sample points p_a, p_b, p_c and p_d with their corresponding subdomains D_a, D_b, D_c and D_d are shown. q_a and q_b are two reference points to establish the convexity in subdomains D_a, D_b and D_c.

In plain words, for every pair of the n subdomains associated with measurement sample points, if there exists a third subdomain lying between them

satisfying (4.11), the convexity of these three subdomains can be established. We shall be reminded that q_a and q_b in (4.11) are not necessarily positions where measurements take place (that is, p_a and p_b), q_a and q_b are introduced as reference points to locate the relevant subdomains for comparison of height values, as Figure 4.3 shows.

4.2.4 Data structure

For the task of automatic trajectory generation we start with the definition of our data structure.

4.2.4.1 TOF-node

We describe the objects in the scene by their height values which can be retrieved at their physical locations. These objects can be registered by the relating local maximum height values. In this sense, a convex-shaped object in the scene can be represented by a tree-like data structure in which the height values are descending.[6] In this tree-like data structure, elements will be differentiated by their associated height values. To this purpose, we define a new type of memory unit called *TOF-node* to store the raw measurement information obtained at a certain sample point and the pointers to the same kind of memory units representing its child nodes. A child TOF-node would refer to the TOF-node associated with the sample point at a relative position of distance 1 (in one dimension) where the height value is descending. These pointers to further TOF-nodes where the height values are descending will be written as W (west, $(\Delta x, \Delta y) = (-1, 0)$), N (north, $(\Delta x, \Delta y) = (0, 1)$), E (east, $(\Delta x, \Delta y) = (1, 0)$) and S (south, $(\Delta x, \Delta y) = (0, -1)$). See Figure 4.4 for an illustration.

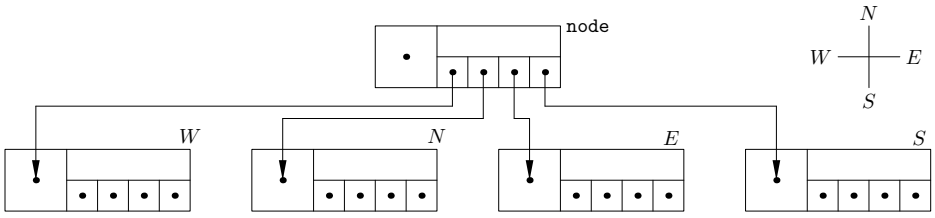

Figure 4.4: Basic structure of the TOF-node. Each TOF-node stores pointers de-referencing up to four child nodes (W, N, E and S). The relative positions of the possible child nodes are given in the compass. Here the TOF-node labelled as "`node`" may or may not have a parent node.

[6] Respecting height values, we request in the implementation a weaker criterion of being "not ascending", cf. Procedure BuildTreeW on page 94. The expansion of the TOF-trees should be continued, as long as it is possible.

It is obvious that a TOF-node, if not associated with a local maximum in height value, has at most three pointers de-referencing further TOF-nodes. Technically, a pointer will be set as "`NULL`", if the target position is not in the observation area Ω, or the height function (due to noise, measurement error etc.) is not defined, or the descending order of the height value cannot be established there.

Consequently, an object with a convex geometric shape can be represented by a TOF-node associated with the local maximum which this object possesses. This is actually what Theorem 4.2 says. In general, although objects in real-world scenarios are not convex in a pure mathematical sense, they can be decomposed into convex objects represented in such a way.

4.2.4.2 TOF-tree

In an image frame, the TOF-node associated with a local maximum will be called a *TOF-tree*. A TOF-tree, interpreted as a TOF-node, will not be de-referenced by other TOF-nodes. TOF-trees in a scene can be constructed in accordance with the local maxima of the height value by the following procedure:

```
   Data: global information
1  repeat
2  |  find point p with maximum height;
3  |  create TOF-node node with p;
4  |  call BuildTree(node);
5  until all points processed;
```

Procedure BUILD

which further invokes the next recursive procedure:

The above procedure expands the TOF-tree under construction in all four directions respecting W, N, E and S. For example, in the direction of W—that is, toward a candidate with a position transition of $(-1, 0)$—the expansion is given by:

Data: global information
Input: `node`

1 allocate memory for W, N, E and S;
2 mark position de-referenced by `node` as processed; `/* this */`
`/* will terminate the recursion of the calling proce- */`
`/* dures of other nodes later */`
3 call BuildTreeW with `node` and candidate position with translation $(-1, 0)$;
4 call BuildTreeN with `node` and candidate position with translation $(0, 1)$;
5 call BuildTreeE with `node` and candidate position with translation $(1, 0)$;
6 call BuildTreeS with `node` and candidate position with translation $(0, -1)$;

Procedure BuildTree(`node`)

Data: global information
Input: `node`, candidate position

1 **if** *position not defined in* Ω **then** `/* condition (*) */`
2 save state "boundary_west"; `/* replace in other di- */`
`/* rections with ''boundary_north'', ''boundary_east'' */`
`/* and ''boundary_south'' respectively */`
3 **else if** *position not already processed and associated height value not ascending* **then** `/* condition (**) */`
4 set TOF-node de-referencing candidate position as `node->`W;
5 call BuildTree(`node->`W);
6 **end**

Procedure BuildTreeW(`node`, candidate position)

The other three procedure components BuildTreeN(`node`, candidate position), BuildTreeE(`node`, candidate position) and BuildTreeS(`node`, candidate position) are analogous.

Technical note. From a perspective of software implementation, Procedures BuildTree, BuildTreeW etc., should be declared and defined as member functions of the class of TOF-tree (or TOF-node). It is for readability's sake that the pseudocode here is presented in a manner of procedural programming rather than object-oriented programming.

By Procedure Build, all the sample points in the scene will be processed, the result of this is a segmentation of the sample points in the observation area Ω. Sample points (cf. technical note in § 4.2.4.1) with extra noises (due to measurement inaccuracy or error etc.) would lead to isolated TOF-trees. These TOF-trees have generally very small sizes and can be discarded. Although later

this results in holes in the objects of the reconstructed scene, the side effect of this phenomenon (for example, in retrieving geometric information of the objects) is in most cases neglectable.

The set of the TOF-trees associated with the local maxima always presents a partitioning of the observation area Ω. In Procedure BuildTreeW of TOF-tree construction, the candidate position must be checked whether it is defined in Ω: in case the condition $(*)$ is met, the expanding TOF-tree reaches the boundary of the geometric setting. We call this boundary type I. If the condition $(**)$ is not met, the expansion of the TOF-tree terminates. In such a case, the candidate position turns out to be a "valley" (lying between two or more convex objects) and thus becomes the boundary of this TOF-tree with others. We call this boundary type II. To specify that this boundary is shared with another TOF-tree T', it can be written as "type II(T')" explicitly. If the condition $(**)$ is met (and implicitly, $(*)$ is not fulfilled), expansion of the TOF-tree continues. The expansion is accompanied with a certain height value H ($H \geq 0$). Expansion regarding a specific H results in what will be called boundary type III and written as "type III(H)". We notice that boundary type III(0) is the contour of the object when it is projected onto the ground floor vertically.

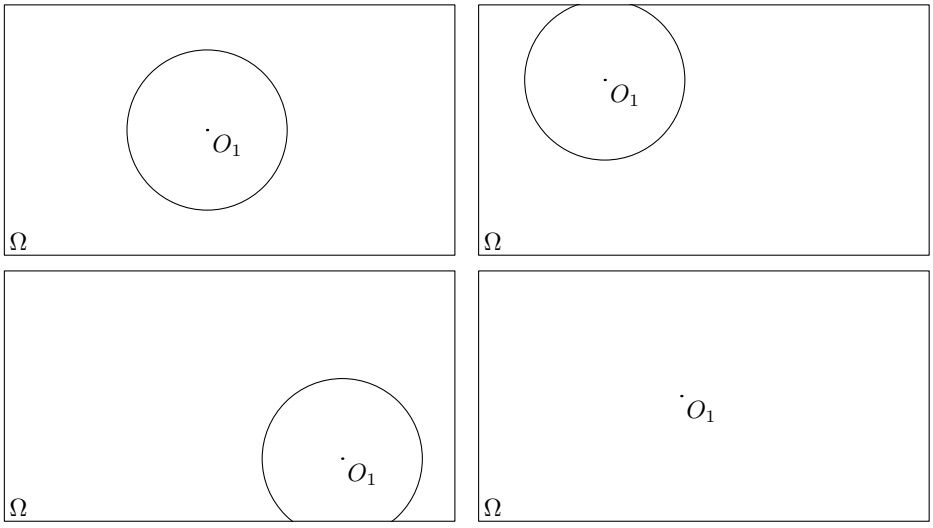

Figure 4.5: Four scenarios of a single object. The root node of a TOF-tree is drawn as a point O_1. Boundary type III(0) is shown in circles. This boundary may or may not be completely visible within Ω.

In Figures 4.5 and 4.6 we present some schematic examples. Figure 4.5 shows the case of a single object O_1. The expansion of the TOF-tree associated with O_1 stops at the boundary of Ω, and is therefore of type I. In the first three subfigures, boundary type III(0) is visible. The last subfigure demonstrates an extreme case in which the would-be boundary type III is not within Ω. This

means, on $\partial\Omega$, the height is above the floor level, this is the case when Ω is completely covered by the object detected by the TOF sensor. A visible but incomplete boundary of type III(0) signals the situation in which an object enters or leaves the observation area.

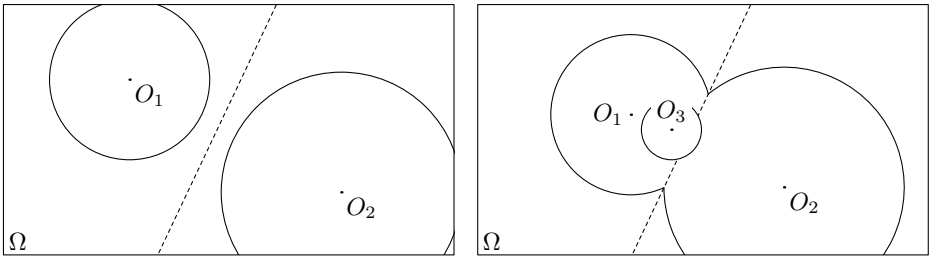

Figure 4.6: Scenarios of two objects. In addition to boundaries type III, boundaries type II are shown in dashed lines. Left: Two separate objects. Right: Two objects with occlusion; one of the objects has two local maxima labelled O_1 and O_3.

Figure 4.6 gives two schematic examples of two objects. Due to the presence of multiple objects, there exist at least two local maxima in height value; the boundary shared by the two objects forms a "valley", which is of type II. The right subfigure of Figure 4.6 presents two objects with occlusion, of which one object is not completely convex and is associated with two local maxima. This object can be represented by the two TOF-trees constructed via these two local maxima.

Technical note. Noise in the measurement leads to the construction of superfluous TOF-trees, since the descending order of the height values can be very often interrupted. Therefore in the implementation, a proper pre-processing (especially smoothing or denoising) of the raw input data is essential for a meaningful (and correct) algorithmic processing.

4.2.5 Basic TOF-tree processing

We consider the problem of retrieving geometric information of a scene at a given height level. This can be seen as "cutting" the objects in the scene through an imaginary plane parallel to the floor. As a result, we get the contour lines of the objects at this height level. In the special case $H = 0$, the contour line is identical to boundary type III(0).

We notice that the topological size of a TOF-tree is the number of the TOF-nodes it contains. When a TOF-tree is "cut" at a given height level H, we may consider this as setting the child node pointers W, N, E and S to `NULL` to terminate the expansion of the corresponding TOF-tree, when the height value H is reached. Thus we observe:

Theorem 4.3. *At any arbitrary height H, the topological size of a TOF-tree equals the area enclosed by its contour line, measured in number of the grid sample points in the original domain Ω.*

Proof. Let `node` de-reference the root node of a TOF-tree. The height value of the root node of this tree will be denoted by `node->H`. Consider the left and right sides of our claim. If `node->H` $< H$, then both sides equal 0. If `node->H` $= H$ and the expansion of the tree structure terminates, both sides equal 1. Otherwise, the left side equals 1 plus the sum of the topological sizes of the tree structures de-referenced by its child nodes `node->`W, `node->`N, `node->`E and `node->`S (a child node with value `NULL` de-references no tree structure anymore). By recursion, these numbers are consistent with the increment (in number of the grid points) of the contour area in the four corresponding directions. □

This proof can be reformulated in the following procedure:

Data: global information
Input: `node`, H

```
1 if node->H < H then
2 |   return 0;
3 else
4 |   call ComputeSize with non-NULL pointers node->W,
  |     node->N, node->E, node->S and H;
5 |   sum the return values;
6 end
7 return the sum plus 1;
```

Procedure ComputeSize(`node`, H)

In a similar way, Procedure ComputeBoundary retrieves the boundary information of a TOF-tree at a particular height level H.

Data: global information
Input: `node`, H

```
1  if node->H < H then
2  |  save state "type III(H)";
3  |  return;
4  end
5  if node->W not NULL then
6  |  call ComputeBoundary with node->W and H;
7  else
8  |  if node de-referenced by node->W is on ∂Ω then
9  |  |  save state "type I";
10 |  else if neighbour node with relative position (Δx, Δy) = (−1, 0)
   |    belongs to another TOF-tree T′ then
11 |  |  save state "type II(T′)";
12 |  end
13 end
14 if node->N not NULL then
15 |  call ComputeBoundary with node->N and H;
16 else
17 |  if node de-referenced by node->N is on ∂Ω then
18 |  |  save state "type I";
19 |  else if neighbour node with relative position (Δx, Δy) = (0, 1)
   |    belongs to another TOF-tree T′ then
20 |  |  save state "type II(T′)";
21 |  end
22 end
23 if node->E not NULL then
24 |  call ComputeBoundary with node->E and H;
25 else
26 |  if node de-referenced by node->E is on ∂Ω then
27 |  |  save state "type I";
28 |  else if neighbour node with relative position (Δx, Δy) = (1, 0)
   |    belongs to another TOF-tree T′ then
29 |  |  save state "type II(T′)";
30 |  end
31 end
32 if node->S not NULL then
33 |  call ComputeBoundary with node->S and H;
34 else
35 |  if node de-referenced by node->S is on ∂Ω then
36 |  |  save state "type I";
37 |  else if neighbour node with relative position (Δx, Δy) = (0, −1)
   |    belongs to another TOF-tree T′ then
38 |  |  save state "type II(T′)";
39 |  end
40 end
```

Procedure ComputeBoundary(`node`, H)

4.2.6 Contour line visualization

Visualization of the contour line of a an image at a particular level is a problem of practical importance. Respecting a TOF-tree, the contour line at the given height level can be retrieved via traversing the TOF-tree. In the general case—as for volume data—the algorithm of marching cubes [42] has been the traditional method of contour generation. Initially designed for the application in three-dimensional cases, marching cubes employ a sophisticated interpolation scheme to construct the contour surface with high resolution. The contour surface generated by this algorithm is composed of a collection of copies of local surface patterns as the result of interpolation of the volume data. Naturally, the algorithm of marching cubes can be modified to apply on surface data in the two-dimensional case. The method presented here is a scanline-like algorithm and hence a completely different approach.

As in the previous paragraphs, let H be a scalar-valued function defined on a two-dimensional grid $\Omega = \{0, \ldots, n_x - 1\} \times \{0, \ldots, n_y - 1\}$. As usual, n_x and n_y denote the width and the height of the grid respectively. In this general purpose method, H is not limited to be the height values measured by a TOF sensor. We further request a minimum size of the grid: $n_x > 2$, $n_y > 2$ ($n_x, n_y \in \mathbb{N}$).

The background idea of our algorithm is to construct a picture C composed of $(n_x - 1)(n_y - 1)$ points by which the change in the scalar values on Ω can be represented. We begin by building a bitarray of $B_{x,y}$,

$$B_{x,y} = \begin{cases} 1, & \text{if } H_{x,y} \geq l, \\ 0, & \text{else,} \end{cases} \tag{4.12}$$

at the given height level l, for $x = 0, \ldots, n_x - 1$, $y = 0, \ldots, n_y - 1$. This picture C will be constructed row by row from $y = 0$ to $n_y - 2$. In each row, the index x runs from 0 to $n_x - 2$. Since the picture C is indeed a discrete scalar-valued function again defined on $(n_x - 1)(n_y - 1)$ points, the $C_{x,y}$-values are determined by the corresponding $B_{x,y}$, $B_{x,y+1}$, $B_{x+1,y}$ and $B_{x+1,y+1}$ from (4.12).

Obviously, there are $2^4 = 16$ possible combinations of the bit values. Accordingly, $C_{x,y}$ can be defined as an integer number representing a state s in the range of $0, \ldots, 15$, after accessing the four corresponding bit values in the original domain Ω:

$$s_{x,y} \Leftarrow B_{x,y} \cdot 1\,, \tag{4.13a}$$

$$s_{x,y} \Leftarrow s_{x,y} + B_{x,y+1} \cdot 2\,, \tag{4.13b}$$

$$s_{x,y} \Leftarrow s_{x,y} + B_{x+1,y} \cdot 4\,, \tag{4.13c}$$

$$s_{x,y} \Leftarrow s_{x,y} + B_{x+1,y+1} \cdot 8\,. \tag{4.13d}$$

Example 4.1. *We demonstrate this idea with a simple example. Consider*

$H_{x,y} = h(f(x), g(y))$, $h(x,y) = x^3 - y^2$, $f(x) = \frac{3x}{n_x - 1}$ *and* $g(y) = \frac{2y}{n_y - 1}$, $n_x = 12$, $n_y = 11$. *For* $l = -\frac{1}{2}$, *we have the following bitarray* B

```
0 0 0 0 0 0 1 1 1 1 1 1    y = 10
0 0 0 0 0 0 1 1 1 1 1 1    y = 9
0 0 0 0 0 1 1 1 1 1 1 1    y = 8
0 0 0 0 0 1 1 1 1 1 1 1    y = 7
0 0 0 0 1 1 1 1 1 1 1 1    y = 6
0 0 0 1 1 1 1 1 1 1 1 1    y = 5
0 0 1 1 1 1 1 1 1 1 1 1    y = 4
1 1 1 1 1 1 1 1 1 1 1 1    y = 3
1 1 1 1 1 1 1 1 1 1 1 1    y = 2
1 1 1 1 1 1 1 1 1 1 1 1    y = 1
1 1 1 1 1 1 1 1 1 1 1 1    y = 0
```

Figure 4.7: Example bitarray B as matrix, x-indices $0, \ldots, 11$ run from left to right.

A simple calculation gives

$$s_{x,y} = \begin{cases} 0, & \text{if } (B_{x,y}, B_{x,y+1}, B_{x+1,y}, B_{x+1,y+1}) = (0,0,0,0), \\ 5, & \text{if } (B_{x,y}, B_{x,y+1}, B_{x+1,y}, B_{x+1,y+1}) = (1,0,1,0), \\ 4, & \text{if } (B_{x,y}, B_{x,y+1}, B_{x+1,y}, B_{x+1,y+1}) = (0,0,1,0), \\ 13, & \text{if } (B_{x,y}, B_{x,y+1}, B_{x+1,y}, B_{x+1,y+1}) = (1,0,1,1), \\ 12, & \text{if } (B_{x,y}, B_{x,y+1}, B_{x+1,y}, B_{x+1,y+1}) = (0,0,1,1), \\ 15, & \text{if } (B_{x,y}, B_{x,y+1}, B_{x+1,y}, B_{x+1,y+1}) = (1,1,1,1). \end{cases} \tag{4.14}$$

Cf. the boxes in Figure 4.7. If we further associate the state variable $s_{x,y}$ *with some simple ASCII characters, the state matrix of* $s_{x,y}$ *can be displayed as an ASCII text block (see Figure 4.8 below and cf. the boxes therein) in which the formation of the contour line can be visually recognized.*

Technical note. (4.14) and Figure 4.8 in Example 4.1 do not address all the 2^4 combinations of $B_{x,y}$, $B_{x,y+1}$, $B_{x+1,y}$ and $B_{x+1,y+1}$. Additionally, for $\begin{smallmatrix}0&0\\1&0\end{smallmatrix}$ (state 1), $\begin{smallmatrix}1&0\\1&1\end{smallmatrix}$ (state 7) $\begin{smallmatrix}0&1\\0&0\end{smallmatrix}$ (state 8) and $\begin{smallmatrix}1&1\\0&1\end{smallmatrix}$ (state 14), the ASCII character "\" can be displayed; for $\begin{smallmatrix}1&0\\0&0\end{smallmatrix}$ (state 2) and $\begin{smallmatrix}1&1\\1&0\end{smallmatrix}$ (state 11), "/" can be displayed; for $\begin{smallmatrix}1&0\\1&0\end{smallmatrix}$ (state 3), "|"; for $\begin{smallmatrix}1&1\\0&0\end{smallmatrix}$ (state 10), "-". For $\begin{smallmatrix}1&0\\0&1\end{smallmatrix}$ (state 6) and $\begin{smallmatrix}0&1\\1&0\end{smallmatrix}$ (state 9), the formation of the contour line cannot be decided locally.

A closer look reveals that for $x = 1, \ldots, n_x - 2$, (4.13a) and (4.13b) are

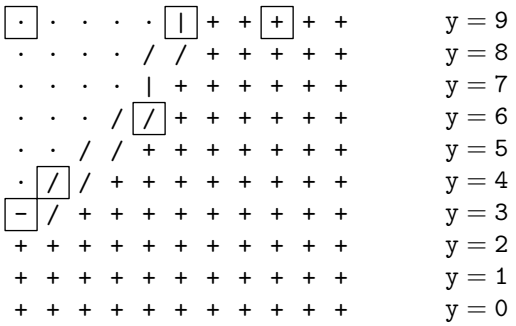

Figure 4.8: ASCII representation of the states of $s_{x,y}$, with "." for $s_{x,y} = 0$, "-" for $s_{x,y} = 5$, "/" for $s_{x,y} = 4, 13$, "|" for $s_{x,y} = 12$ and "+" for $s_{x,y} = 15$.

functionally equivalent (in high-level programming languages) to

$$s_{x,y} \Leftarrow s_{x-1,y} \gg 2\,, \tag{4.15}$$

where "$\gg$" denotes the bitwise shift operator.

In addition, we present an abstract colour scheme for the contour. It is obvious that no local contour exists if the four grid positions have the same value in B ($\boxed{\begin{smallmatrix}0&0\\0&0\end{smallmatrix}}$ and $\boxed{\begin{smallmatrix}1&1\\1&1\end{smallmatrix}}$, $s_{x,y} = 0, 15$). Overall, we assign an indexed colour value to the pixel in C at (x, y):

$$C_{x,y} = \begin{cases} 1, & s_{x,y} = 1, 2, 4, 7, 8, 11, 13, 14, \\ \frac{2}{3}, & s_{x,y} = 3, 5, 10, 12, \\ \frac{1}{3}, & s_{x,y} = 6, 9, \\ 0, & s_{x,y} = 0, 15. \end{cases} \tag{4.16}$$

In (4.16), the cases $s_{x,y} = 1, 2, 4, 7$ represent the reverse of those of $s_{x,y} = 14, 13, 11, 8$ respectively (the same is with $3, 5$ to $12, 10$ and 6 to 9 and 0 to 15). In these cases, the contour line will go through the current pixel in C; hence, $C_{x,y}$ is assigned with the full colour value 1. The cases $s_{x,y} = 3, 5, 10, 12$ are less ambiguous than $s_{x,y} = 6, 9$ and $C_{x,y}$ is given a higher value ($\frac{2}{3}$ in comparison to $\frac{1}{3}$). If C is to be a bitmap, (4.16) can be further simplified into

$$C_{x,y} = \begin{cases} 1, & s_{x,y} = 1, \ldots, 14, \\ 0, & s_{x,y} = 0, 15, \end{cases} \tag{4.17}$$

which gives a clear clue when a pixel should be coloured in C. A real-world example of this is presented in Figures 4.9 and 4.10. The specific TOF hardware

in use was a SwissRanger4000 sensor[7] with a resolution of $176 \cdot 144$.

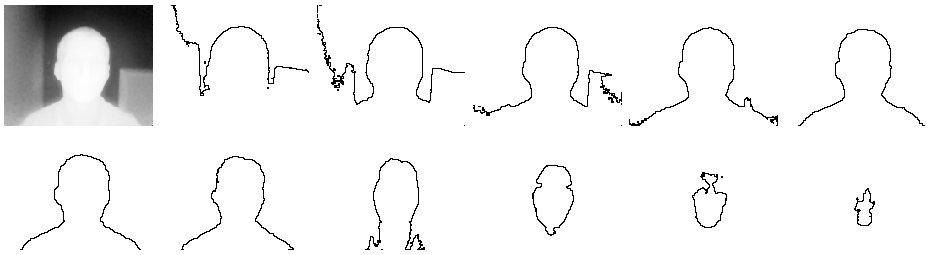

Figure 4.9: The original image H in a linear gray scale and its contours on different height levels. The simplified colour scheme of (4.17) has been applied for the individual pixels.

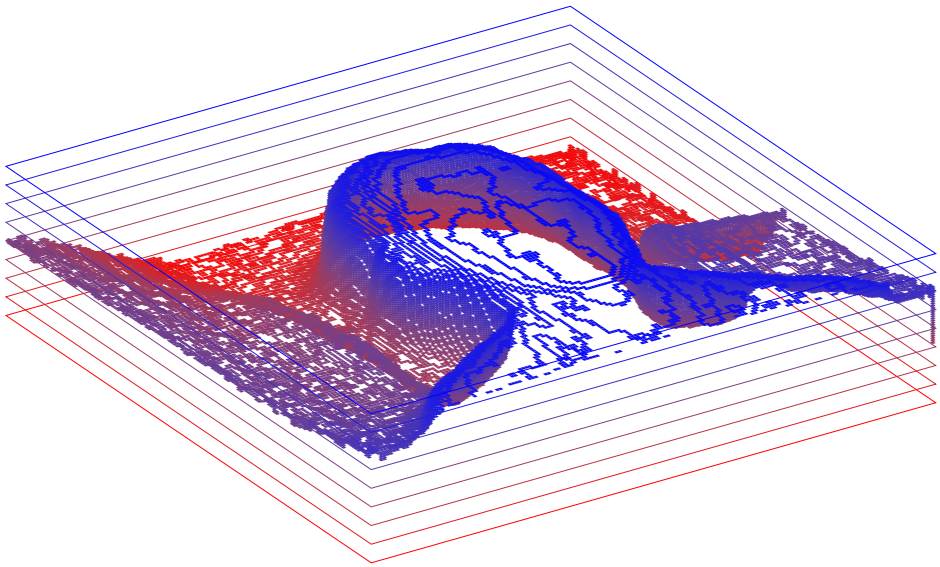

Figure 4.10: The contours of H in a three-dimensional frame. The colour evolution indicates the linear transition of the height levels. The simplified colour scheme of (4.17) has been applied for the individual pixels.

Technical note. We notice that C and H have different image sizes. However, as long as H is not too small, this effect can be largely neglected in the visualization process. On the other hand, since the generated contour line is composed of unstructured two-dimensional grid points only, we may not apply some of the common post-processing techniques on the enclosed contour area directly. Nevertheless, owing to the bitwise shift operation in (4.15) (which reduces roughly half of the computational costs), the algorithm itself is very efficient for visualization. In fact, (4.13a)–(4.13d) offer nothing but a binary encoding of the state $s_{x,y} = (B_{x+1,y+1}B_{x+1,y}B_{x,y+1}B_{x,y})_2$ which makes the bitwise shift operation in (4.15) possible.

[7]Product of Mesa Imaging AG (Switzerland), homepage `http://www.mesa-imaging.ch`.

4.3 An application in TOF

In this section we present a practical application of the TOF technology. A specific TOF sensor iris-DIST500[8][9] has been in use as industrial application of passenger counting in public transportation in Berlin. Here the same input data captured by the iris-DIST500 sensor will be used to track the passengers as moving objects; from the automatically generated trajectories the counting procedure can be thus realized. We shall later provide a detailed comparison with the result delivered by the original device. We start with a sequence of n input frames.

4.3.1 Frame-based processing

Technically speaking, a frame F_i $(i = 0, \ldots, n-1)$ is an image, consisting of the complete measurement information defined in (4.7), created by the sensor at a certain time. The segmentation of the sample points of a single input frame F_i has been detailed in § 4.2.4.2. The result of this step is a collection of TOF-trees $t_{i,0}, \ldots, t_{i,f_i-1}$, whereas f_i is the number of the TOF-trees constructed in the frame F_i, for $i = 0, \ldots, n-1$. The variable i serves as a time index. Basic processing of these trees has been given in § 4.2.5.

4.3.1.1 TOF-object

We observe that objects detected in the scene can be represented by sets of connected TOF-trees. Let $T_i = \{0, \ldots, f_i - 1\}$ be the set of the TOF-tree indices in frame F_i. Let further 2^{T_i} denote the power set of the index set T_i. For every nonempty element B of 2^{T_i} ($B \subseteq T_i$, $B \neq \emptyset$), we examine the boundary information of the TOF-trees associated with this index set B: when these are connected, we will construct a candidate object with these indices and call it simply *TOF-object*. By "being connected" we mean that either B is composed of exactly one element; or if B is composed of two elements, there exists a common boundary shared by the TOF-trees associated with these two indices; or otherwise for every index pair (p, q) with $p, q \in B$, $p \neq q$, there exists a series of elements $\alpha, \ldots, \omega$ from the rest of B—that is, $\{\alpha, \ldots, \omega\} \subseteq B \setminus \{p, q\}$—so that all the pairs $(p, \alpha), \ldots, (\omega, q)$ are connected as in the case of a subset B with exactly two elements. As a union of the TOF-trees associated with the indices from the set B, a TOF-object serves as a candidate for a real object in the scene. The geometric information of a TOF-object can be easily retrieved once we know the corresponding TOF-tree indices.

[8] Product of iris-GmbH (Germany), homepage `http://www.irisgmbh.de`.

[9] In our previous work [11], it had been mistakenly reported to be a SwissRanger4000 sensor. In comparison to the former ansatz, the method presented in the current text contains significant improvements.

Remark: We notice the exponential complexity incurred by the power set 2^{T_i}, so the pre-processing (smoothing etc.) of the original input data is very necessary. In addition, filtering out irrelevant TOF-trees is essential as well, this is the case when their sizes are below a pre-defined threshold value, since sample points of poor measurement lead to isolated TOF-trees. Empirically speaking, the growth of the computational costs is much lower than being exponential (in terms of TOF-object construction), since very often a large part of the elements of 2^{T_i} are not qualified for the construction of TOF-objects, because the corresponding TOF-trees are not connected. On the other hand—in view of software implementation—we do not really need to generate the power set 2^{T_i} in every frame: a sufficiently large power set $2^{T_{\max}}$ (for example, $T_{\max} = \{0, \ldots, f_{\max} - 1\}$, $f_{\max}$ denotes the largest number of TOF-trees in the frame sequence) can be used in all frames, this will reduce the computational costs substantially.

4.3.1.2 TOF-trajectory

Now we consider another data structure which we will call *TOF-trajectory*. A TOF-trajectory is the record of a series of TOF-objects in a continuous sequence of frames. Given such a record, the physical trajectory (that is, position transition) of the corresponding object detected in the scene can be reconstructed.

Data: O_{i+1}, L_i
mark incomplete TOF-trajectories from L_i as active;
compute D_{i+1}; `/* part 1 */`
repeat `/* part 2 */`
  find minimum $d_{i+1,u,v}$ from D_{i+1} with active $o_{i+1,u}$ and $l_{i,v}$;
  if $d_{i+1,u,v}$ *below threshold* **then**
    append $o_{i+1,u}$ to $l_{i,v}$;
    deactivate $l_{i,v}$;
  else
    construct a new incomplete TOF-trajectory by $o_{i+1,u}$ for frame F_{i+1};
  end
  for all $u' = 0, \ldots, |O_{i+1}| - 1$ that $o_{i+1,u} \cap o_{i+1,u'} \neq \emptyset$ deactivate $o_{i+1,u'}$; `/* including especially the case u = u' */`
until *no active item in* D_{i+1} *left*;
if *there are active TOF-trajectories left* **then** `/* part 3 */`
  `/* there should be no active TOF-object left */`
  mark all active TOF-trajectories as complete;
else
  `/* there should be no active TOF-trajectory left */`
  construct new incomplete TOF-trajectories by the TOF-trees contained in the remaining active TOF-objects;
end

Procedure MATCH

In a start frame F_i ($i = 0, \ldots n-1$), when there exists no active TOF-trajectory from the previous frame F_{i-1},[10] we may initialize a series of new (and thus incomplete) TOF-trajectories $l_{i,0}, \ldots, l_{i,f_i-1}$ by the present TOF-trees in F_i, f_i stands for the number of TOF-trees in frame F_i as defined previously. In the subsequent frame F_{i+1}, a TOF-trajectory $l_{i,v}$ will be marked as active, if $l_{i,v}$ contains a TOF-object from the previous frame F_i, otherwise this TOF-trajectory will be concluded (and thus complete).

Assume, in frame F_{i+1}, we have constructed a collection of TOF-objects $O_{i+1} = \left\{o_{i+1,0}, \ldots, o_{i+1,|O_{i+1}|-1}\right\}$; and at the same time, we have a collection of active TOF-trajectories $L_i = \left\{l_{i,0}, \ldots, l_{i,|L_i|-1}\right\}$ from the previous frame F_i. We can construct a distance matrix D_{i+1} with $|O_{i+1}| \cdot |L_i|$ items:

$$D_{i+1} = \left(d_{i+1,u,v}\right), \quad \text{for } u = 0, \ldots, |O_{i+1}| - 1, v = 0, \ldots, |L_i| - 1, \tag{4.18}$$

where $d_{i+1,u,v}$ is the distance from the TOF-object $o_{i+1,u}$ in frame F_{i+1} to the active TOF-trajectory $l_{i,v}$ from frame F_i. The distance function to compute $d_{i+1,u,v}$ will be discussed in the next section.

We can match the current TOF-objects with the active TOF-trajectories and update the latter for frame F_{i+1}, as described in Procedure MATCH. This procedure has three parts. The first part is the initialization and computation of a distance matrix D_{i+1} combining all current TOF-objects in frame F_{i+1} and active TOF-trajectories from frame F_i. Next, we search for the minimum among all active items in the distance matrix after every update of the TOF-objects and the TOF-trajectories (the distance matrix D_{i+1} itself does not need to be re-calculated). Given a successful search result of a TOF-object and a TOF-trajectory, we either, if the current minimum distance is below an empirical threshold value, append the TOF-object to the TOF-trajectory, or we construct a new TOF-trajectory with the TOF-object. The updated TOF-trajectory and the new constructed TOF-trajectory will be deactivated temporarily for frame F_{i+1}. Line 11 says that once a TOF-object has been used to construct or update a TOF-trajectory, all the other TOF-objects with which there is an overlapping of the TOF-trees must be removed from the distance matrix, or in other words, these TOF-objects become inactive. The third part of this procedure deals with the remaining TOF-objects or TOF-trajectories, but not both (otherwise there would be active $d_{i+1,u,v}$ items present in the distance matrix D_{i+1}). The remaining TOF-trajectories we have here will be concluded (i. e. complete), since no TOF-objects are appended to them, they will not be activated in the subsequent frame. The remaining TOF-objects will be used to construct new TOF-trajectories, since they are not matched with any active TOF-trajectory in frame F_{i+1}. Procedure MATCH runs through the sequence of the frames $F_1, \ldots, F_{n-1}$.

[10] When $i - 1 \geq 0$; if $i = 0$, there will be no TOF-trajectory in frame F_0 (a TOF-trajectory is a record of TOF-objects in at least two successive frames).

4.3.2 Distance scoring by the Kalman filter

4.3.2.1 Definition of distance

Owing to the optical distortion (mentioned on page 88) of the iris-DIST500 sensor, a rigid object detected in the scene may not have a constant size at different recording positions. Let F_i be the current frame to be processed. Let the last TOF-object contained in trajectory $l_{i,v}$ be written as $o_{i,k}$. For the item $d_{i+1,u,v}$ in (4.18), therefore, in addition to the position prediction $\hat{\mathbf{x}}_{i+1,k}$, a size prediction $\hat{s}_{i+1,k}$ of the TOF-object $o_{i,k}$ will be considered as well. For the distance matrix (4.18) we define:

$$d_{i+1,u,v} = |\mathbf{x}_{i+1,u} - \hat{\mathbf{x}}_{i+1,k}|_2 \cdot s^p_{i+1,u,v}, \tag{4.19}$$

where

$$s_{i+1,u,v} = \frac{\max(s_{i+1,u},\ \hat{s}_{i+1,k})}{\min(s_{i+1,u},\ \hat{s}_{i+1,k})},$$

and p is a positive weight control parameter. For a technical reason (to avoid division by zero), we request that the predicted size $\hat{s}_{i+1,k}$ has at least the size of one pixel to evaluate $s_{i+1,u,v}$:

$$\hat{s}_{i+1,k} \Leftarrow \max(1, \hat{s}_{i+1,k}).$$

Two additional threshold values θ_d and θ_s can be applied to validate the estimate of $d_{i+1,u,v}$:

$$|\mathbf{x}_{i+1,u} - \hat{\mathbf{x}}_{i+1,k}|_2 \leq \theta_d, \quad s_{i+1,u,v} \leq \theta_s,$$

cf. line 5 of Procedure Match.

4.3.2.2 A very short introduction to the Kalman filter

Next, we consider the problem of position and size prediction. The Kalman filter [34] gives an elegant solution to these kind of problems formulated in linear dynamical systems. For a better readable text we refer to [16]. With a Kalman filter, the state transition in a system is assumed to have the following form:

$$\mathbf{x}_{i+1} = \mathbf{F}\mathbf{x}_i + \mathbf{B}\mathbf{u}_{i+1} + \mathbf{w}_{i+1},$$

where $\mathbf{x}$ refers to the state vector in the system[11]; $\mathbf{F}$ stands for the state transition matrix; $\mathbf{u}$ is the input control vector; $\mathbf{B}$ is the input control matrix which delivers impact from the elements of the input control vector to the state vector; $\mathbf{w}$ embodies the term of noise in the state vector. The start point of the Kalman filter is to observe that $\mathbf{w}$ obeys a zero-mean (multivariate) normal distribution

[11] We adopt the same symbol for state vector as in common literature; however, this is not the same with the position vector $\mathbf{x}$ in (4.19).

with covariance governed by matrix $\mathbf{Q}$. The measurement equation of the system is:

$$\mathbf{z}_{i+1} = \mathbf{H}\mathbf{x}_{i+1} + \mathbf{v}_{i+1},$$

where $\mathbf{z}$ is the vector composed of the corresponding measurements; $\mathbf{H}$ is the transition matrix mapping the state vector into the measurement domain; $\mathbf{v}$ is the vector of measurement noise terms and obeys a second zero-mean normal distribution with covariance $\mathbf{R}$. In general, $\mathbf{F}$, $\mathbf{B}$, $\mathbf{H}$, $\mathbf{Q}$ and $\mathbf{R}$ are not limited to be time-constant.

Furthermore, we have the estimate $\hat{\mathbf{x}}$ and its covariance matrix $\mathbf{P}$. The computation of the Kalman filter is composed of two parts. In the stage of prediction (without formal deduction), we have:

$$\begin{aligned} \hat{\mathbf{x}}'_{i+1} &= \mathbf{F}\hat{\mathbf{x}}_i + \mathbf{B}\mathbf{u}_{i+1}, \\ \mathbf{P}'_{i+1} &= \mathbf{F}\mathbf{P}_i\mathbf{F}^T + \mathbf{Q}. \end{aligned} \tag{4.20}$$

In the stage of update (without formal deduction), we have:

$$\begin{aligned} \hat{\mathbf{x}}_{i+1} &= \hat{\mathbf{x}}'_{i+1} + \mathbf{K}_{i+1}\left(\mathbf{z}_{i+1} - \mathbf{H}\hat{\mathbf{x}}'_{i+1}\right), \\ \mathbf{P}_{i+1} &= \left(\mathbf{I} - \mathbf{K}_{i+1}\mathbf{H}\right)\mathbf{P}'_{i+1}, \end{aligned} \tag{4.21}$$

where $\mathbf{I}$ is the unit matrix and $\mathbf{K}_{i+1}$ is the so-called *Kalman gain* at the current moment:

$$\mathbf{K}_{i+1} = \mathbf{P}'_{i+1}\mathbf{H}^T\left(\mathbf{H}\mathbf{P}'_{i+1}\mathbf{H}^T + \mathbf{R}\right)^{-1}.$$

$\mathbf{K}$ is in fact the covariance of the fusion—as a new Gaussian distribution, which is the essence of the Kalman filter—of the prediction distribution and the measurement distribution. The superscript $'$ in (4.20) and (4.21) indicates that $\hat{\mathbf{x}}_{i+1}$ and $\mathbf{P}_{i+1}$ are in an intermediate state of the current estimate, they are often called *prior* values, whereas after the measurement update of (4.21), $\hat{\mathbf{x}}_{i+1}$ and $\mathbf{P}_{i+1}$ become *posterior*.

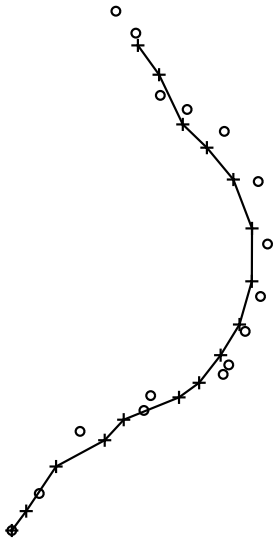

Figure 4.11: An example of prediction by the Kalman filter

The computation of (4.20) and (4.21) will be carried out alternately with measurement $\mathbf{z}$ serving as input correction for the update every time. Now we consider a series of random position transitions. Planar positions will be measured in this test case. Let

$$\mathbf{x} = \begin{pmatrix} x \\ y \\ v_x \\ v_y \end{pmatrix}$$

denote the state vector, consisting of coordinates x and y and speed components v_x and v_y, then the state transition matrix can be easily given as:

$$\mathbf{F} = \begin{pmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

with Δt being the time lapse between two measurements. Naturally, we have

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

to map state vector $\mathbf{x}$ back into the measurement domain.

If we oversimplify our model, we can assume zero input control, that is, $\mathbf{B} = \mathbf{0}$ and $\mathbf{u} = \mathbf{0}$.

The initialization of $\mathbf{P}$, $\mathbf{Q}$ and $\mathbf{R}$ requires *a priori* knowledge, if nonexistent, their initialization can be performed by guess. According to [59], larger covariances usually lead to faster convergence (if at all).

This simple model is depicted in Figure 4.11 in which the measured positions are shown in "+" and the predicted positions in "∘". The same setting has been applied in the position prediction of our TOF input data.

4.3.3 Test results and further discussion

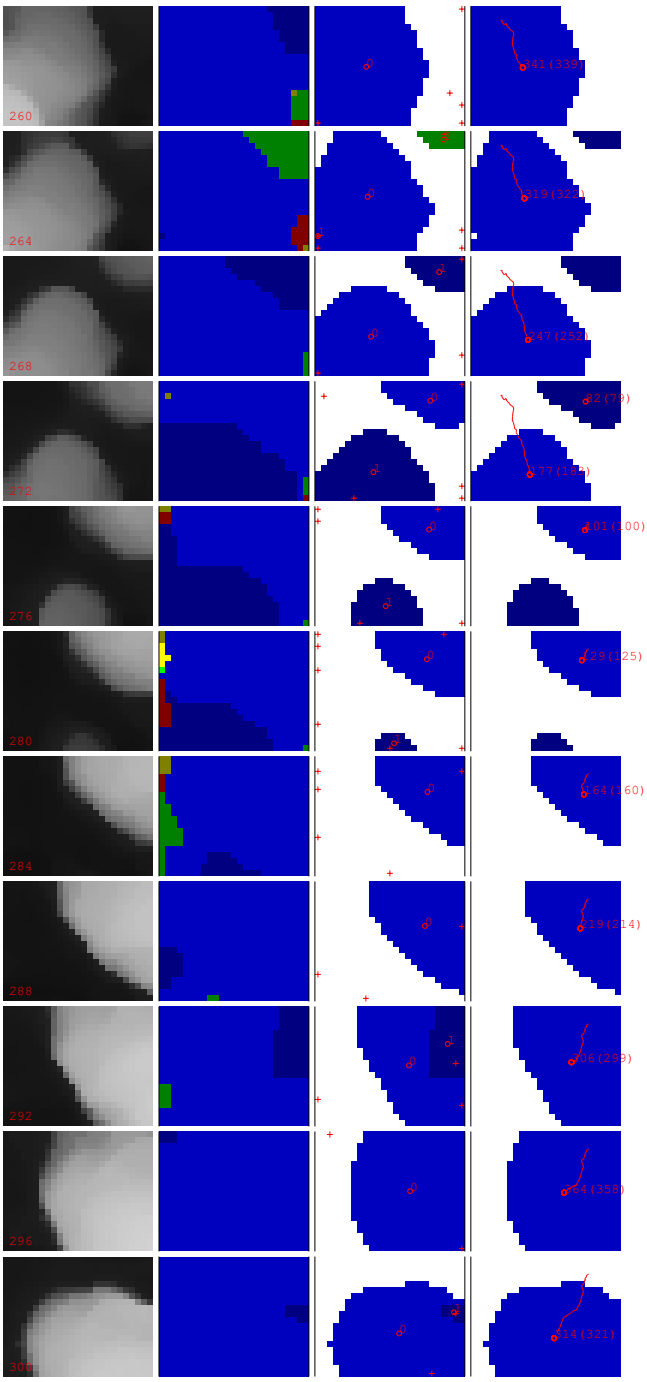

Figure 4.12: Test example 1

Test results of our application on a collection of data records captured by a DIST500 sensor will be presented in the following.

The DIST500 sensor has a relatively low resolution of $25 \cdot 20$ ($n_x = 25$, $n_y = 20$), the diagonal coverage is about 60°. Data recording can be performed at a frequency of 16 frames per second. The device had been designed to capture and count passenger movement along the y-axis of the observation area.

In Figures 4.12 and 4.13 two test examples are presented. Unfortunately, respecting the DIST500 sensor as a proprietary industrial device, no specification about its installation or its recording function is available. This means that even the exact physical size of the observation area is unknown.

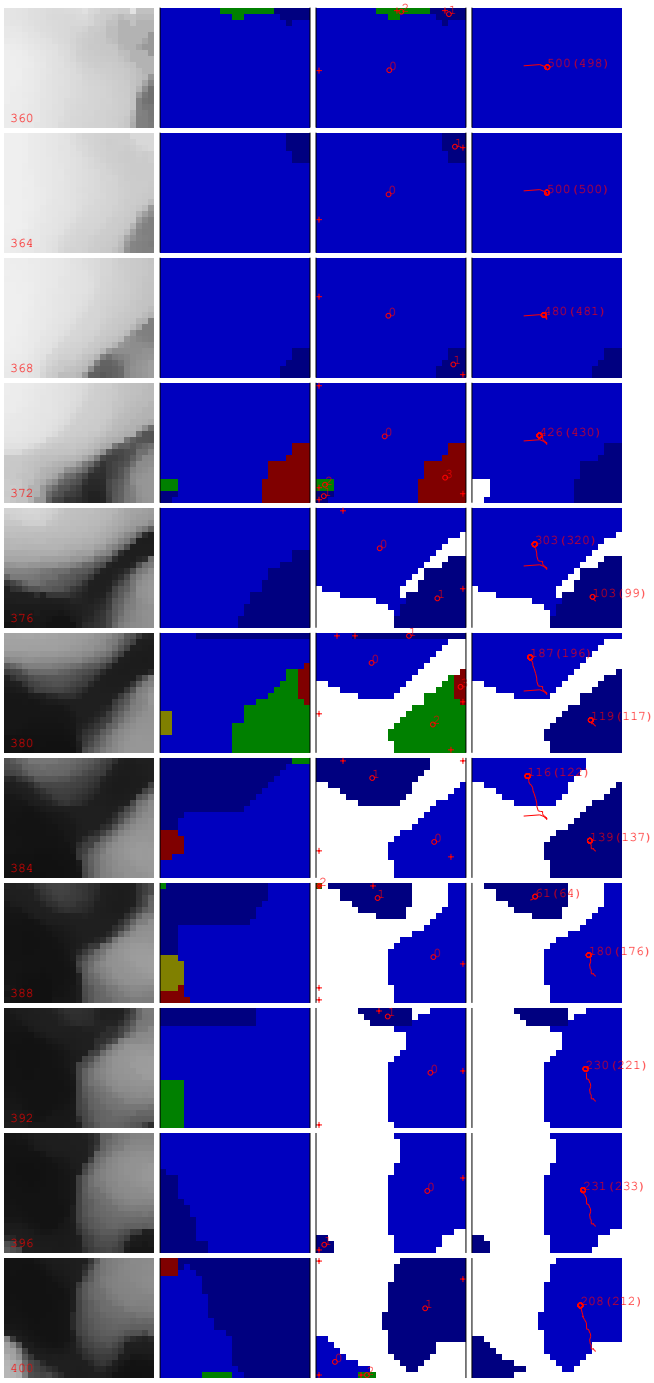

Figure 4.13: Test example 2

In fact, if adequate installation information (that is, h_0 in § 4.2.2) about the sensor is available, aided by the knowledge of the recording frame frequency, the Euclidean distance threshold θ_d can be estimated. In the tests, we have set $\theta_d = 0.2 \cdot n_y$ (that is, 4 pixels) and $p = 0.5$ (cf. (4.19)). In addition to the already mentioned optical distortion, owing to this sensor's limited coverage, objects in the scene can often be captured only partially. Therefore, the same object may appear with quite different sizes in different frames. Experiments showed $\theta_s = 1.25$ to be a good choice (see discussion later).

In these two figures (it is recommended to zoom in on the electronic version of this text to see the details), the height values have been presented in the first column as gray scale images. For a better editorial effect, the frames are selected at an equal distance of 4 from the original sequence. The time indices are given in the images of the first column. The segmentation of the frames by means of TOF-trees is given in the second column. The third column presents a special cutting plane, and on this plane, TOF-trees representing pos-

sible TOF-objects as candidates for the detected passengers are drawn in various colours. The cutting plane has been defined to emulate a height of $h = 1.4\,\mathrm{m}$ in the world coordinate system. Positions of the root nodes (that is, of maximum heights) of the corresponding TOF-trees are shown in "+", whereas their centroids are shown in "∘". It can be easily identified that the root node and the centroid of a TOF-tree have generally different positions. TOF-trees below the cutting plane are not drawn. In the fourth column, we give the possible trajectories of the detected objects, "∘" stands for the predicted position at the current moment, the size (in pixels) of the current object is given as well (the number in parentheses refers to the predicted size in pixels). Both figures deal with the case of the encountering of two closely walking people. (Owing to the rather poor date quality, sometimes it is not easy to identify the scene even manually.) In Figure 4.13, one of the objects had been very close to the recording sensor and, in some frames, had virtually covered the complete observation angle of the sensor. The extracted trajectories can be verified in the next figure.

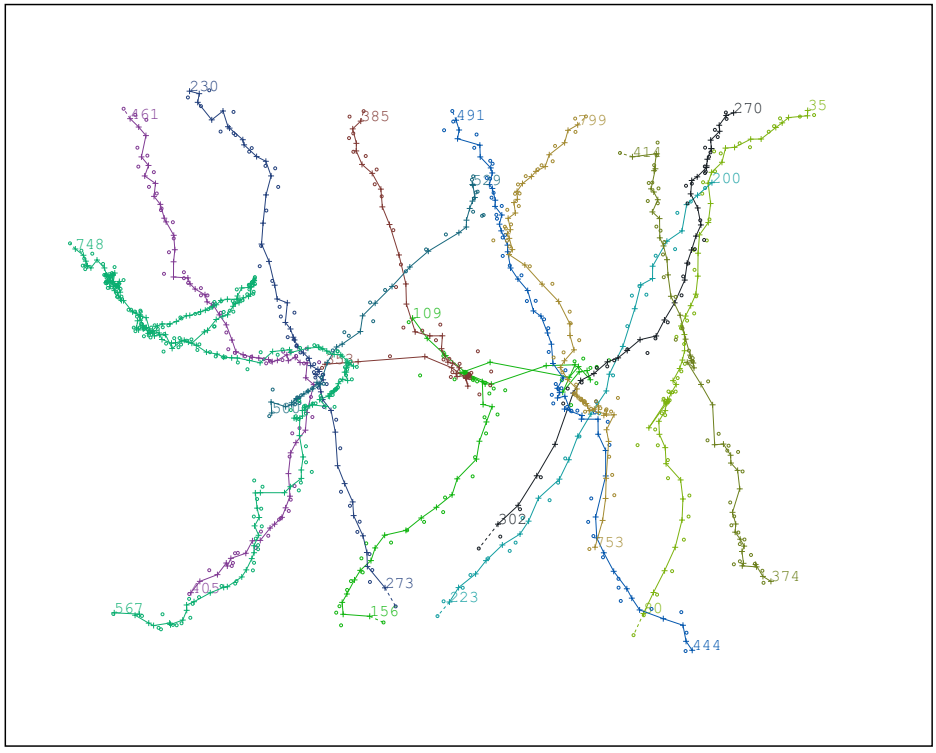

Figure 4.14: Detected trajectories from a sequence of input frames as a complete test example. In addition to the start and end frame indices, measurement points and predicted positions are shown in "+" and "∘" respectively.

In Figure 4.14, the trajectories are labelled with the corresponding start and end frame indices. For example, the trajectory with label 230–273 refers to the first trajectory recorded in Figure 4.12 (which is last shown in the frame with

time index 272). The second trajectory is labelled 270–302. The two trajectories in Figure 4.13 are labelled with 353–385 and 374–414 respectively. If we look carefully, we will see in Figure 4.13 a third but very short trajectory (in frame with time index 388). This trajectory came into being because the main trajectory (with label 353–385) had been concluded too early. Irregular trajectories of this kind do not really affect the final counting result, since they can be filtered out by length. However, without proper a priori knowledge of the parameters of the recording sensor and the environment, often valid trajectory are decomposed into various short trajectories. To solve this problem, we may go through all the generated trajectories to see whether certain linkages (via position check, for example) can be enabled to recover the original trajectories. An adjusted counting can be performed after this recovery step.

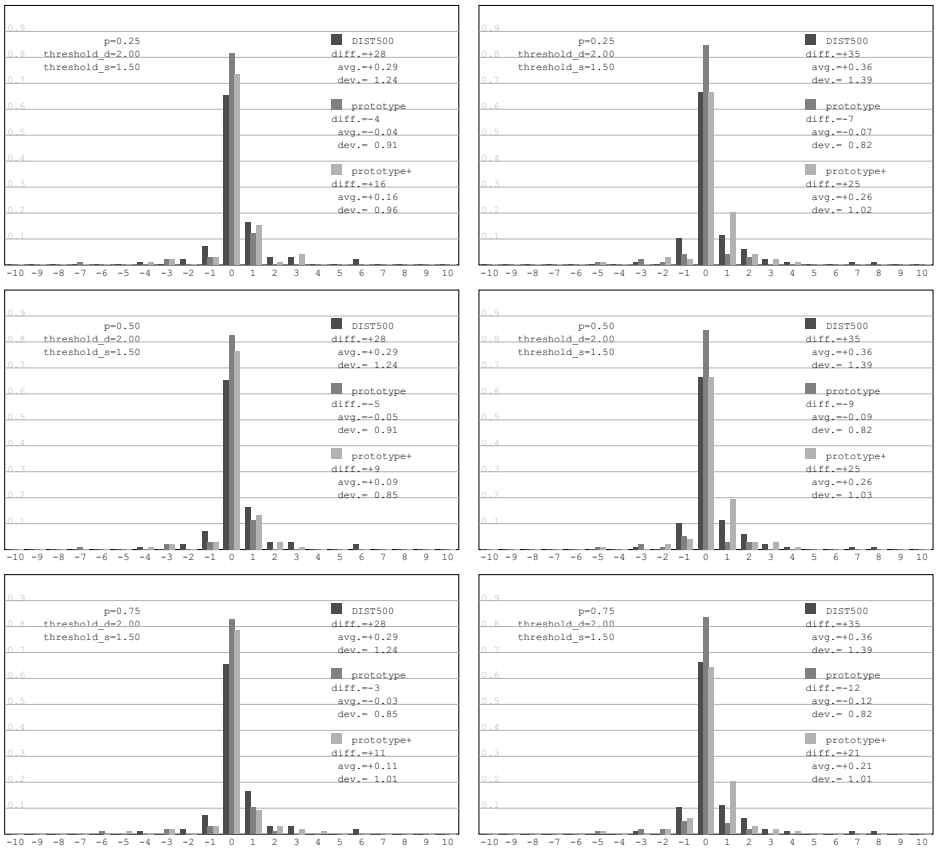

Figure 4.15: Complete test result part 1. Horizontal axis: Counting errors (in numbers). Vertical axis: Occurrence of the errors, shown in a relative scale. "`diff.`" refers to the total error summed over multiple input data records with its average value "`avg.`" and standard deviation "`dev.`". Parameters in the test: $p \in \{0.25, 0.5, 0.75\}$, $\theta_d = 0.1 \cdot n_y$, $\theta_s = 1.5$.

In Figures 4.15–4.18, we list the test result comparison of our application

with a DIST500 sensor. The left and right columns refer to the counting results in the two walking directions respectively. Deviations from manually verified counting results (the latter might still contain slight errors owing to the poor quality of the data recordings) are shown. Our application is labelled "`prototype`" and "`prototype+`", denoting the procedures without and with adjusted counting via possible compensation for lost or broken trajectories, respectively. It is to be mentioned that alone the error (deviation in number) respecting a data record gives no qualitative estimate of the counting procedure, since input data records can be of very different time lengths (varying from seconds to hours). In general, the total error ("`diff.`") serves as a good indicator for the counting quality.

Empirically, $[0.25, 0.75]$ appears to be a reasonable calibration range for p.

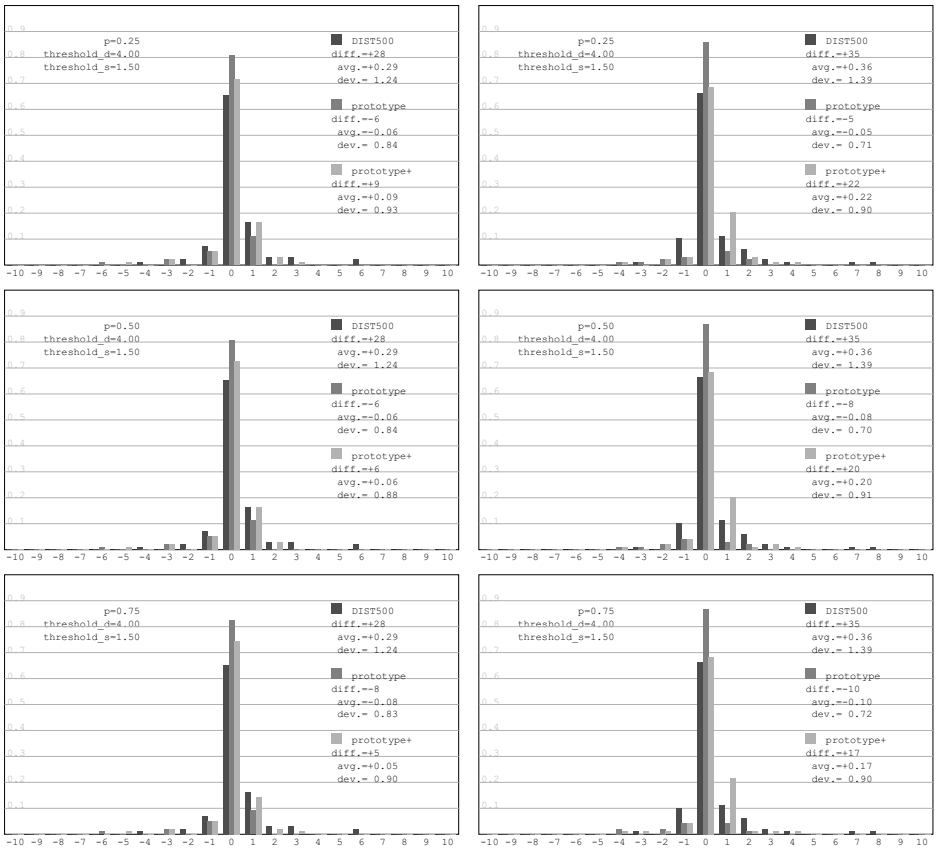

Figure 4.16: Complete test result part 2. Parameters in the test: $p \in \{0.25, 0.5, 0.75\}$, $\theta_d = 0.2 \cdot n_y$, $\theta_s = 1.5$.

Figures 4.15 and 4.16 show that with the current settings, $\theta_s = 1.5$ produces rather poor results; on the other hand, experiments have shown that $\theta_s = 1.125$ or another similar value configuration would deliver even poorer results (figures not listed). From our experience, $\theta_s = 1.25$ is a good choice.

In Figures 4.17 and 4.18 test results of the parameter settings of $p \in \{0.25, 0.5, 0.75\}$, $\theta_s = 1.25$ with $\theta_d \in \{0.1 \cdot n_y, 0.2 \cdot n_y\}$ are given. Application "`prototype+`" (that is, with adjusted counting compensation) combined with $p \in \{0.5, 0.75\}$ and $\theta_d \in \{0.1 \cdot n_y, 0.2 \cdot n_y\}$ offers substantial improvement to the original DIST500 sensor. The best result has been achieved by $p = 0.75$, $\theta_d = 0.2 \cdot n_y$ (see last row of Figure 4.18), the overall errors in both directions are -1 and 1 respectively. In general, hardware-relevant information would ease the parameter calibration[12] greatly.

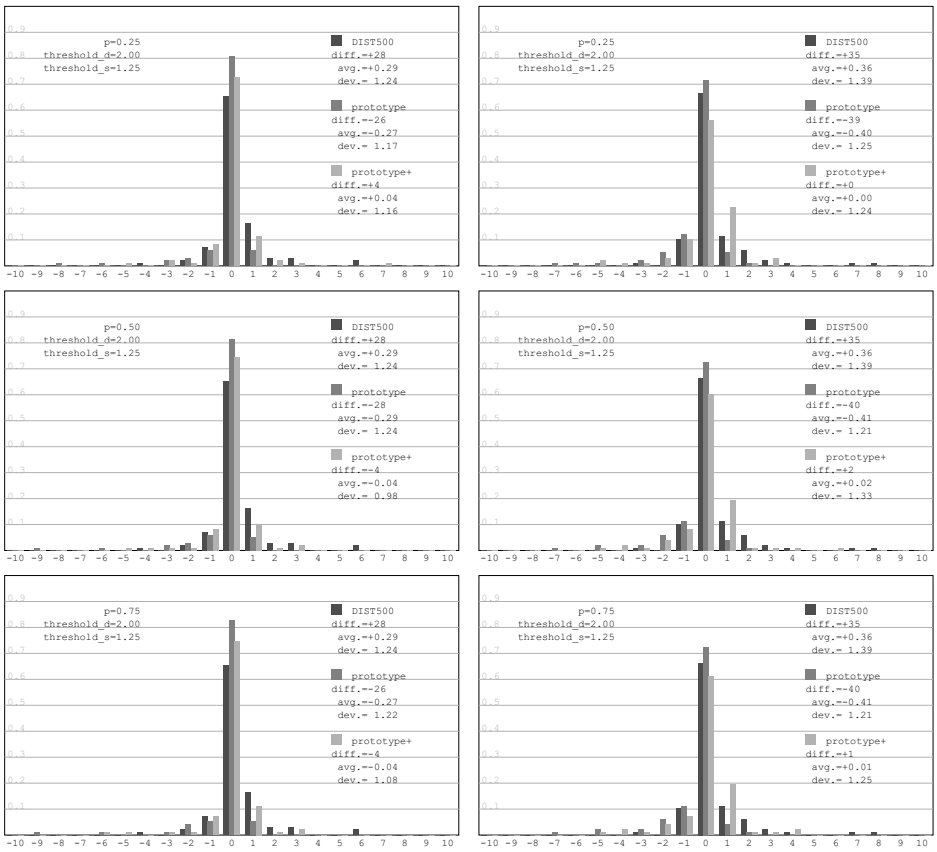

Figure 4.17: Complete test result part 3. Parameters in the test: $p \in \{0.25, 0.5, 0.75\}$, $\theta_d = 0.1 \cdot n_y$, $\theta_s = 1.25$.

By means of a reliable counting application, automatic capture and generation of trajectories can be realized. Although not included in this current text, pattern recognition can (and should) be applied in our frame-based processing in the future development of the so-called "intelligent sensor"s. In a broader sense, applying our method, data recorded by any TOF sensors, although these may

[12]For θ_d, see explanation on page 110, θ_s depends on the observation angle of the sensor, p tends to be an empirical control parameter.

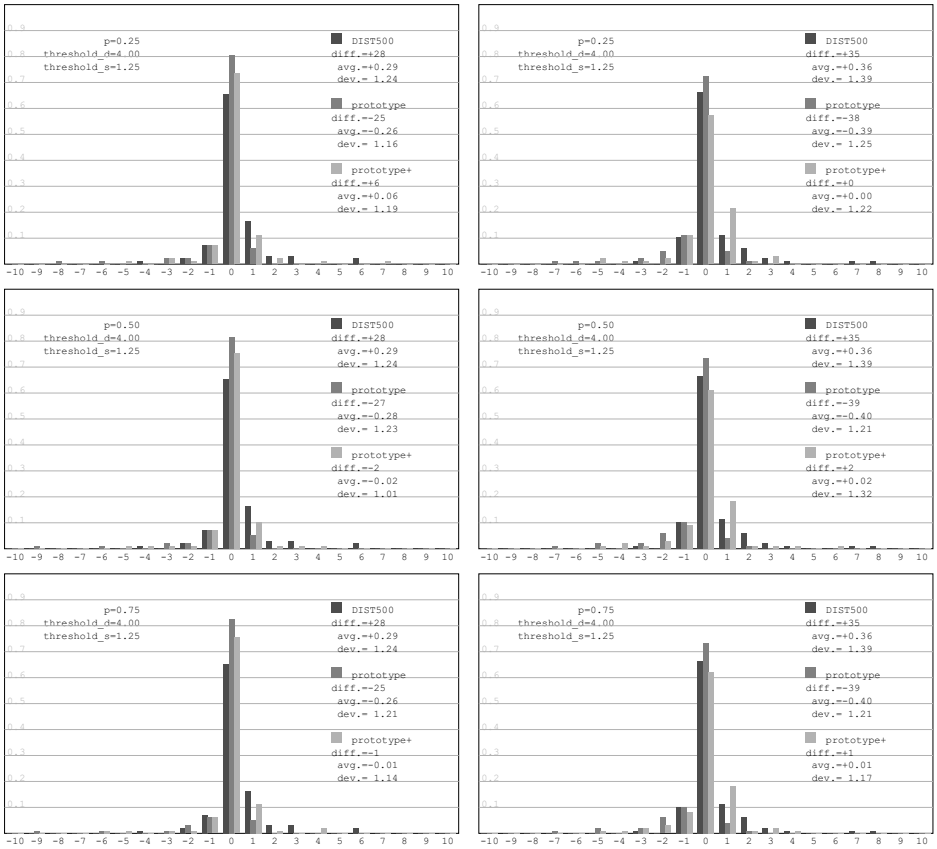

Figure 4.18: Complete test result part 4. Parameters in the test: $p \in \{0.25, 0.5, 0.75\}$, $\theta_d = 0.2 \cdot n_y$, $\theta_s = 1.25$.

have different capture capabilities in terms of size of the observation area and recording frequency etc., can be used to detect and generate trajectories of moving objects. Reconstructed spacial information of pedestrians as moving objects can be of great value for the calibration of existing simulation models; this applies particularly to our models presented in the last two chapters. The photogrammetric method presented in § 4.1, although it is based on a completely different principle, can be employed to serve the same purpose as well.

Appendix
Notes on Supplementary Materials

The software implementation of the topics studied in the current text has been submitted in electronic form separately. The implementation has been grouped into "project"s. The projects have been developed in the NetBeans[13] IDE (Integrated Development Environment) on the Linux platform. Configuration files relating to the IDE have been provided, although without these, programs still can be opened and modified in all common editors. All programs have been carefully debugged (unfortunately as in the usual case the author is unable to provide a formal guarantee of their being bug-free); all C++ programs have been developed to pass the well-known Valgrind[14] test.

A.1 Project `contour.nb`

This C++ project refers to § 4.2.6 and has the following simple structure:

```
contour.nb/
├── data ..................................... input data directory
│   └── frame.bin ..... an example input file saved as a binary stream
├── main.cpp ..... main driver program to test the above example input
├── Makefile
├── marching.cpp
├── marching.h
├── nbproject/ ..... NetBeans configuration directory
├── output/ ..... directory for output files
└── simple_test.cpp ..... a simple test program of a user-defined
                          function of two variables; the modifiable
                          function definition is given in the macro
                          SIMPLE_FUNC in the program
```

The project can be built by issuing the command in the project directory `contour.nb/`:

```
$ make
```

Figure 4.10 has been produced by running the executable `./main`.

[13] Homepage `https://netbeans.org`, current version 8.2.

[14] An advanced error detection and profiling tool for Linux executables, homepage `http://www.valgrind.org`, version in use 3.10.0.

A.2 Project `multilane.nb`

This C++ project refers to § 2.3 and has the following structure:

```
multilane.nb/
├── aclocal.m4
├── autogen.sh ..... main bootstrap script
├── auxil/ ..... directory of auxiliary functions
│   ├── decls.h
│   ├── error_manip.c
│   ├── error_manip.h
│   ├── file_manip.c
│   ├── file_manip.h
│   ├── Makefile.am
│   ├── Makefile.in
│   ├── random.cpp
│   └── random.h
├── config.h.in
├── configure ..... main configuration script
├── configure.ac ..... main input file for Autoconf
├── definition.h
├── exec/ ..... directory of driver programs and executables
│   ├── acceleration.h
│   ├── asymm_symm.cpp
│   ├── coefficient.h
│   ├── main_.h ..... this header file contains the control variables of the simulation; for the modification of these variables to take effect, a new build (compilation) by make is necessary
│   ├── main.h
│   ├── Makefile.am
│   ├── Makefile.in
│   ├── model1_diagram.cpp ..... driver program to generate system-wide fundamental diagrams, to be conditionally compiled
│   ├── model1_main.cpp ..... driver program to generate lane-based fundamental diagrams, to be conditionally compiled
│   ├── model1_main.h
│   └── model1_test.cpp ..... driver program to generate space-time diagrams, to be conditionally compiled
├── install-sh
├── Makefile.am ..... main input file for Automake
├── Makefile.in
└── model/ ..... directory of model components
    ├── Makefile.am
    ├── Makefile.in
    ├── model1.cpp
    ├── model1.h
    ├── multilane.cpp
    └── multilane.h
```

```
    ├── site.cpp
    ├── site.h
    ├── vehicle.cpp
    └── vehicle.h
├── nbproject/ ..... NetBeans configuration directory
└── output/ ..... directory for output files
```

The user may start the `configure` script to generate the `Makefile`s for the compilation of the project, the binary executables will reside in the `exec/` directory:

```
$ ./configure
$ make
```

The binaries are to be executed in the project directory.

Technical note. This project adopts the GNU build system of Autoconf[15] and Automake[16]. In fact, the main configuration script `configure` is generated by the command `autoreconf` in the main bootstrap script `autogen.sh` in the project directory. Possible configuration options will be detailed by issuing the command `./configure -h`.

Technical note. In directory `exec/`, conditional compilation has been applied to reduce code redundancy (which is unnecessary and considered error-prone). Conditional compilation of `model1_test.cpp` results in executables `model1_test` (cf. Figures 2.5–2.7), `model1_symm` (cf. Figure 2.8 and 2.9), `model1_dist2` (cf. Figure 2.10), `model1_accel`, `model1_seq` and `model1_revseq` (the latter three are not used in the current text). Conditional compilation of `model1_main.cpp` results in `model1_main` (cf. Figures 2.12 and 2.13) and `model1_main_reconf` (not used in the current text). Conditional compilation of `model1_diagram.cpp` results in `model1_diagram` (cf. Figure 2.14), `model1_diagram_reconf` (cf. Figures 2.16 and 2.17), `model1_diagram_reconf_accel` (cf. Figure 2.18) and `model1_diagram_accel` (not used in the current text).

A.3 Project `pgrid.nb`

The first part of this JAVA[17] project refers to § 3.1.3–3.1.6. The second part refers to § 3.2.4. The project has the following structure:

```
pgrid.nb/
├── build/ ..... NetBeans build directory
├── build.xml ..... NetBeans build configuration file
└── config/ ..... directory of static and dynamic user configuration inputs
```

[15] Homepage `https://www.gnu.org/software/autoconf/autoconf.html`.
[16] Homepage `https://www.gnu.org/software/automake/automake.html`.
[17] Homepage `http://www.oracle.com/technetwork/java/javase/overview/index.html`.

- `dist/` — NetBeans distribution directory
- `manifest.mf`
- `nbproject/` — NetBeans configuration directory
- `output/` — output directory
- `src/` — source code directory
  - `mjc/` — source code package
    - `Base.java` — basic system and user definitions
    - `Cell.java` — base class for grid cells in the models
    - `conf/` — directory of functions to handle dynamic configuration inputs
      - `DynamicInput.java`
      - `DynamicRecord.java`
      - `misc/`
        - `Translation.java`
    - `ModelFrame.java`
    - `ModelFramePanel.java`
    - `Model.java` — base class for models
    - `Overlay.java` — class for density overlay
    - `Particle2D.java`
    - `Particle.java` — base class for particles in the models
    - `path3/` — directory of model implementation in path-oriented coordinate system
      - `Configuration.java`
      - `MainPath3.java` — demonstration program for modelling in path-oriented coordinate system
      - `model/` — directory of modelling components in path-oriented coordinate system
        - `ArcPath3.java`
        - `CellPath3.java`
        - `ModelPath3Frame.java`
        - `ModelPath3FramePanel.java`
        - `ModelPath3.java`
        - `ParticlePath3.java`
        - `TransitionPath3.java`
    - `Position2D.java`
    - `reg4/` — directory of model implementation on rectangular grids
      - `Approximation.java`
      - `ArcApproximation.java` — class to simulate visibility angle
      - `Atomic.java` — definition for elementary steps
      - `aux/`
        - `MakeLookup.java` — program to machine-generate the lookup tables (see `Lookup.java` below)
      - `bresenham/`
        - `Algorithm.java`
        - `BresenhamTest.java`
        - `CircleMidpoint.java`
      - `Configuration.java`
      - `Lookup.java` — lookup tables for elementary steps
      - `Main1Reg4.java` — demonstration program of a simple test case in modelling on a rectangular grid

```
├── Main2Reg4.java ..... demonstration program for modelling
│                        on a rectangular grid, with configura-
│                        tion input for model validation
├── MainPlayer.java ..... demonstration program to reconstruct
│                        graphic output from a configuration in-
│                        put
├── model/ ..............directory of modelling components on
│   │                    rectangular grids
│   ├── CellReg4.java
│   ├── GridReg4.java
│   ├── GridReg4Test.java ..........a simple test program for
│   │                               local density calculation
│   ├── Model1Reg4.java
│   ├── Model2Reg4.java
│   ├── ModelReg4Frame.java
│   ├── ModelReg4FramePanel.java
│   ├── ModelReg4.java
│   ├── ParticleReg4.java
│   ├── Player.java
│   ├── StaticInputLoader.java
│   ├── TransitionReg4.java
│   └── TransitionReg4Test.java
└── Model2Reg4MultipleTest.java .... main program for model
                                     validation on a rectangular
                                     grid, without graphic out-
                                     put
└── TextDisplay.java
```

After building the project, the user can run the main programs directly in the NetBeans IDE. Running the binaries in command line is possible in the project directory `pgrid.nb/`, for example:

```
$ java -classpath build/classes/ mjc.reg4.Main1Reg4
```

where `build/classes/` is the directory in which the complied JAVA classes are stored, which has to be specified by the `-classpath` option.

A.4 Project `trajectory.nb`

This C++ project refers to § 4.3 and has the following structure:

```
trajectory.nb/
├── autogen.cmake ..................................main bootstrap script
└── auxil/ ..........................................directory of auxiliary functions
    ├── CMakeLists.txt
    ├── file_manip.c
    ├── file_manip.h
    └── irregularity.c
```

```
    ├── irregularity.h
├── build/ ..... local build directory
├── clean.cmake ..... main build script I
├── CMakeLists.txt ..... main input file for CMake
├── compile.cmake ..... main build script II
├── configuration.xml ..... user-defined parameter configuration file
├── definition.h ..... main header file
├── device/ ..... directory of device-specific functions
│   ├── CMakeLists.txt
│   ├── dist500png.c
│   ├── dist500png.h
│   ├── engine.cpp
│   ├── engine.h
│   ├── input_loader.cpp
│   ├── input_loader.h
│   ├── paths.cpp
│   ├── paths.h
│   ├── z_correction.c
│   └── z_correction.h
├── exec/ ..... directory of driver and test programs
│   ├── CMakeLists.txt
│   ├── configuration.cpp
│   ├── configuration.h
│   ├── global_variables.h
│   ├── main.cpp
│   ├── parser.h
│   ├── test_configuration.cpp
│   ├── test_engine.cpp
│   ├── test_frame.cpp
│   ├── test_kalman1.cpp
│   ├── test_kalman.cpp
│   ├── test_matrix.cpp
│   └── test_powerset.cpp
├── extra/ ..... directory of secondary computing modules and functions
│   ├── CMakeLists.txt
│   ├── kalman_filter1.cpp
│   ├── kalman_filter1.h
│   ├── kalman_filter.cpp
│   ├── kalman_filter.h
│   ├── matrix.c
│   ├── matrix.h
│   ├── powerset_generation.cpp
│   └── powerset_generation.h
├── nbproject/ ..... NetBeans configuration directory
├── output/ ..... directory for PNG and PostScript output files
└── tof/ ..... directory of primary computing modules and functions
    ├── CMakeLists.txt
    ├── TOF_boundary.cpp
    ├── TOF_boundary.h
    ├── TOF_distance_matrix.cpp
    └── TOF_distance_matrix.h
```

```
├── TOF_frame.cpp
├── TOF_frame.h
├── TOF_node.cpp
├── TOF_node.h
├── TOF_object_candidate.cpp
├── TOF_object_candidate.h
├── TOF_pixel.cpp
├── TOF_pixel.h
├── TOF_position.cpp
├── TOF_position.h
├── TOF_sequence.cpp
├── TOF_sequenceCV.cpp
├── TOF_sequence.h
├── TOF_trajectory.cpp
├── TOF_trajectory.h
├── TOF_tree.cpp
└── TOF_tree.h
```

This project requires the libraries libpng[18], Boost[19] and OpenCV[20]. These three libraries are contained as packages in all major Linux distributions.

This project applies the CMake[21] build system. The user may start the main bootstrap script `autogen.sh` to build the project. In fact, `autogen.sh` which contains two further scripts: `clean.cmake` and `compile.cmake`; to build the project with existing `Makefiles`, `compile.cmake` would be enough. The binaries in the `build/exec/` directory are to be executed in the project directory `trajectory.nb/`. The `input/` directory, which contains 98 original datasets, must be placed in the same directory as the project directory. Running `build/exec/test_engineCV -h` gives detailed usage information of the demonstration program. A simplified demonstration program without graphic output is provided by `build/exec/test_engine`. For the main program `build/exec/main`, file `input.csv`, which stores the original counting results, is to be placed in the parent directory of the project as well.

[18] Homepage `http://www.libpng.org/pub/png/libpng.html`, version in use 1.2.50.
[19] Homepage `http://www.boost.org`, version in use 1.55.0.
[20] Homepage `http://opencv.org`, tested versions 3.1.0 and 2.4.9.
[21] Homepage `https://cmake.org`.

Bibliography

[1] A. Adamatzky, editor. *Game of Life cellular automata*. Springer-Verlag London Limited, 2010. ISBN 978-1-84996-216-2.

[2] G. Antonini, M. Bierlaire, and M. Weber. Discrete choice models of pedestrian walking behavior. *Transportation Research Part B*, 40:667–687, 2006.

[3] L. Asher, M. Aresu, E. Falaschetti, and J. Mindell. Most older pedestrians are unable to cross the road in time: a cross-sectional study. *Age and Ageing*, 41(5):690–694, 2012.

[4] C. Bays. Introduction to cellular automata and Conway's Game of Life. In Adamatzky [1], pages 1–7.

[5] M. Bertalmío, G. Sapiro, and G. Randall. Morphing active contours. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(7):733–737, 2000.

[6] P. J. Besl. Active, optical range imaging sensors. *Machine Vision and Applications*, 1(2):127–152, 1988.

[7] J. E. Bresenham. Algorithm for computer control of a digital plotter. *IBM Systems Journal*, 4(1):25–30, 1965.

[8] T. J. Broida and R. Chellappa. Estimation of object motion parameters from noisy images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(1):90–99, 1986.

[9] C. Burstedde, K. Klauck, A. Schadschneider, and J. Zittartz. Simulation of pedestrian dynamics using a two-dimensional cellular automaton. *Physica A*, 295:507–525, 2001.

[10] M.-J. Chen, G. Bärwolff, and H. Schwandt. A study of step calculations in traffic cellular automaton models. In *13th International IEEE Conference on Intelligent Transportation Systems*, pages 747–752, 2010. doi: 10.1109/ITSC.2010.5625180.

[11] M.-J. Chen, G. Bärwolff, and H. Schwandt. Time-Of-Flight technology applied in pedestrian movement detection. In Daamen et al. [14], pages 189–194. doi: 10.1016/j.trpro.2014.09.028.

[12] M.-J. Chen, G. Bärwolff, and H. Schwandt. A deductive model for multilane vehicular traffic. In *18th IEEE International Conference on Intelligent Transportation Systems*, pages 900–905, 2015. doi: 10.1109/ITSC.2015.151.

[13] K. Creath. Phase-measurement interferometry techniques. *Progress in Optics*, 26:349–393, 1988.

[14] W. Daamen, D. C. Duives, and S. P. Hoogendoorn, editors. *The Conference on Pedestrian and Evacuation Dynamics 2014*, 2014. Elsevier B. V.

[15] E. W. Dijkstra. Cooperating sequential processes. *E. W. Dijkstra Archive*, EWD123, 1968.

[16] R. Faragher. Understanding the basis of the Kalman filter via a simple and intuitive derivation. *IEEE Signal Processing Magazine*, 29(5):128–132, 2012.

[17] M. Frank, M. Plaue, H. Rapp, U. Köthe, B. Jähne, and F. A. Hamprecht. Theoretical and experimental error analysis of continuous-wave time-of-flight range cameras. *Optical Engineering*, 48(1):013602, 2009.

[18] V. García-Morales. From deterministic cellular automata to coupled map lattices. *Journal of Physics A: Mathematical and Theoretical*, 49(29):295101, 2016.

[19] C. Gloor, P. Stucki, and K. Nagel. Hybrid techniques for pedestrian simulations. In Sloot et al. [66], pages 581–590.

[20] L. Gray. A mathematician looks at Wolfram's New Kind of Science. *Notices of the AMS*, 50(2):200–211, 2003.

[21] M. Hansard, S. Lee, O. Choi, and R. Horaud. *Time-of-Flight Cameras: Principles, Methods and Applications*. Springer London Heidelberg New York Dordrecht, 2013. ISBN 978-1-4471-4657-5.

[22] D. Helbing. Theoretical foundation of macroscopic traffic models. *Physica A*, 219:375–390, 1995.

[23] D. Helbing. *Verkehrsdynamik: Neue physikalische Modellierungskonzepte*. Springer-Verlag Berlin Heidelberg, 1997. ISBN 978-3-642-63834-3.

[24] D. Helbing and P. Molnár. Social force model for pedestrian dynamics. *Physical Review E*, 51(5):4282–4286, 1995.

[25] D. Helbing, I. Farkas, and T. Vicsek. Simulating dynamical features of escape panic. *Nature*, 407:487–490, 2000.

[26] D. Helbing, A. Johansson, and H. Z. Al-Abideen. Dynamics of crowd disasters: An empirical study. *Physical Review E*, 75:046109, 2007.

[27] S. P. Hoogendoorn and P. H. L. Bovy. State-of-the-art of vehicular traffic flow modelling. *Journal of Systems and Control Engineering*, 215(4):283–303, 2001.

[28] S. P. Hoogendoorn and P. H. L. Bovy. Pedestrian route-choice and activity scheduling theory and models. *Transportation Research Part B*, 38:169–190, 2004.

[29] B. K. P. Horn and B. G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1980.

[30] R. L. Hughes. A continuum theory for the flow of pedestrians. *Transportation Research Part B*, 36:507–535, 2002.

[31] F. Huth, G. Bärwolff, and H. Schwandt. A macroscopic multiple species pedestrian flow model based on heuristics implemented with finite volumes. In Weidmann et al. [72], pages 585–601. ISBN 978-3-319-02446-2. doi: 10.1007/978-3-319-02447-9_49.

[32] L. Isaksen and H. J. Payne. Freeway traffic surveillance and control. *Proceedings of the IEEE*, 61(5):526–536, 1973.

[33] A. S. Jalal and V. Singh. The state-of-the-art in visual object tracking. *Informatica*, 36(3):227–248, 2012.

[34] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[35] J. Kari. Theory of cellular automata: A survey. *Theoretical Computer Science*, 334:3–33, 2005.

[36] M. Keller, J. Orthmann, A. Kolb, and V. Peters. A simulation framework for Time-Of-Flight sensors. In *2007 International Symposium on Signals, Circuits and Systems*, volume 1, pages 125–128, 2007.

[37] A. Keßel, H. Klüpfel, J. Wahle, and M. Schreckenberg. Microscopic simulation of pedestrian crowd motion. In Schreckenberg and Sharma [61], pages 193–200. ISBN 978-3-540-42690-5.

[38] R. Kühne. Greenshields' legacy: Highway traffic. In *Transportation Research Circular E-C149: 75 Years of the Fundamental Diagram for Traffic Flow Theory: Greenshields Symposium*, pages 3–10. Transportation Research Board, 2011.

[39] G. Lämmel and M. Plaue. Getting out of the way: Collision-avoiding pedestrian models compared to the real world. In Weidmann et al. [72], pages 1275–1289. ISBN 978-3-319-02446-2. doi: 10.1007/978-3-319-02447-9_105.

[40] R. Lange and P. Seitz. Solid-state Time-of-Flight range camera. *IEEE Journal of Quantum Electronics*, 37(3):390–397, 2001.

[41] M. J. Lighthill and G. B. Whitham. On kinematic waves. II. A theory of traffic flow on long crowded roads. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 229(1178):317–345, 1955.

[42] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics*, 21(4):163–169, 1987.

[43] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of Imaging Understanding Workshop*, pages 121–130, 1981.

[44] A.-R. Mansouri. Region tracking via level set PDEs without motion computation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(7):947–961, 2002.

[45] M. Moussaïd, D. Helbing, and G. Theraulaz. How simple rules determine pedestrian behavior and crowd disasters. *Proceedings of the National Academy of Sciences of the United States of America*, 108(17):6884–6888, 2011.

[46] B. Murgante, S. Misra, M. Carlini, C. M. Torre, H.-Q. Nguyen, D. Taniar, B. O. Apduhan, and O. Gervasi, editors. *Computational Science and Its Applications – ICCSA 2013*, volume 7975 of *Lecture Notes in Computer Science*, 2013. Springer-Verlag Berlin Heidelberg. ISBN 978-3-642-39639-7.

[47] K. Nagel and M. Schreckenberg. A cellular automaton model for freeway traffic. *J. Phys. I France*, 2:2221–2229, 1992.

[48] K. Nagel, D. E. Wolf, P. Wagner, and P. Simon. Two-lane traffic rules for cellular automata: A systematic approach. *Physical Review E*, 58(2): 1425–1437, 1998.

[49] J. von Neumann. *Theory of Self-Reproducing Automata*. University of Illinois Press, 1966. Edited and completed by A. W. Burks.

[50] C. Niclass, P. Besse, and E. Charbon. Arrays of single photon avalanche diodes in CMOS technology: Picosecond timing resolution for range imaging. In *Proceedings of the First Range Imaging Research Day at ETH*, 2005.

[51] C. Niclass, M. Sergio, and E. Charbon. A single photon avalanche diode array fabricated in deep-submicron CMOS technology. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 81–86, 2006.

[52] K. Nishinari, A. Kirchner, A. Namazi, and A. Schadschneider. Extended floor field CA model for evacuation dynamics. *IEICE Transactions on Information and Systems*, E87-D(3):726–732, 2004.

[53] T. Oggier, M. Lehmann, R. Kaufmann, M. Schweizer, M. Richter, P. Metzler, G. Lang, F. Lustenberger, and N. Blanc. An all-solid-state optical range camera for 3D real-time imaging with sub-centimeter depth resolution (SwissRanger$^{\mathrm{TM}}$). *Proc. SPIE*, 5249, Optical Design and Engineering: 534–545, 2004.

[54] M. Plaue, M.-J. Chen, G. Bärwolff, and H. Schwandt. Trajectory extraction and density analysis of intersecting pedestrian flows from video recordings. In Stilla et al. [69], pages 285–296. ISBN 978-3-642-24392-9. doi: 10.1007/978-3-642-24393-6_24.

[55] M. Plaue, M.-J. Chen, G. Bärwolff, and H. Schwandt. Multi-view extraction of dynamic pedestrian density fields. *Photogrammetrie, Fernerkundung, Geoinformation*, 5:547–555, 2012. doi: 10.1127/1432-8364/2012/0138.

[56] T. M. Rengarasu, H. N. Jayawansa, and G. P. W. Perera. Estimation of pedestrian walking speeds at controlled cross walks in Sri Lanka – a pilot study. In *Proceedings of International Symposium on Advances in Civil and Environmental Engineering Practices for Sustainable Development (ACEPS 2012)*, pages 91–95, 2012.

[57] P. I. Richards. Shock waves on the highway. *Operations Research*, 4(1): 42–51, 1956.

[58] M. Rickert, K. Nagel, M. Schreckenberg, and A. Latour. Two lane traffic simulations using cellular automata. *Physica A*, 231:534–550, 1996.

[59] J. Le Roux. An introduction to Kalman filtering: Probabilistic and deterministic approaches, 2003.

[60] A. Schadschneider. Cellular automaton approach to pedestrian dynamics – Theory. In Schreckenberg and Sharma [61], pages 75–85. ISBN 978-3-540-42690-5.

[61] M. Schreckenberg and S. D. Sharma, editors. *Pedestrian and Evacuation Dynamics*, 2002. Springer-Verlag Berlin Heidelberg. ISBN 978-3-540-42690-5.

[62] H. Schwandt, F. Huth, G. Bärwolff, and S. Berres. A multiphase convection-diffusion model for the simulation of interacting pedestrian flows. In Murgante et al. [46], pages 17–32. ISBN 978-3-642-39639-7. doi: 10.1007/978-3-642-39640-3_2.

[63] R. Schwarte, H.-G. Heinol, Z. Xu, and K. Hartmann. New active 3D vision system based on rf-modulation interferometry of incoherent light. *Proc. SPIE*, 2588, Intelligent Robots and Computer Vision XIV: Algorithms, Techniques, Active Vision, and Materials Handling:126–134, 1995.

[64] A. Seyfried, B. Steffen, and T. Lippert. Basics of modelling the pedestrian flow. *Physica A*, 368(1):232–238, 2006.

[65] B. W. Silverman. *Density estimation for statistics and data analysis*. Chapman and Hall, 1986. ISBN 0-412-24620-1.

[66] P. M. A. Sloot, B. Chopard, and A. G. Hoekstra, editors. *Proceedings of the 6th International Conference on Cellular Automata for Research and Industry, ACRI 2004*, volume 3305 of *Lecture Notes in Computer Science*, 2004. Springer-Verlag Berlin Heidelberg. ISBN 978-3-540-23596-5.

[67] T. Spirig, P. Seitz, O. Vietze, and F. Heitger. The lock-in CCD—Two-dimensional synchronous detection of light. *IEEE Journal of Quantum Electronics*, 31(9):1705–1708, 1995.

[68] B. Steffen and A. Seyfried. Methods for measuring pedestrian density, flow, speed and direction with minimal scatter. *Physica A*, 389(9):1902–1910, 2010.

[69] U. Stilla, F. Rottensteiner, H. Mayer, B. Jutzi, and M. Butenuth, editors. *Photogrammetric Image Analysis, ISPRS Conference, PIA 2011*, volume 6952 of *Lecture Notes in Computer Science*, 2011. Springer-Verlag Berlin Heidelberg. ISBN 978-3-642-24392-9.

[70] F. van Wageningen-Kessels, H. van Lint, K. Vuik, and S. Hoogendoorn. Genealogy of traffic flow models. *EURO Journal on Transportation and Logistics*, 4(4):445–473, 2015.

[71] N. Waldau, P. Gattermann, H. Knoflacher, and M. Schreckenberg, editors. *Pedestrian and Evacuation Dynamics 2005*. Springer-Verlag Berlin Heidelberg, 2007. ISBN 978-3-540-47062-5.

[72] U. Weidmann, U. Kirsch, and M. Schreckenberg, editors. *Pedestrian and Evacuation Dynamics 2012*, 2014. Springer International Publishing Switzerland. ISBN 978-3-319-02446-2.

[73] S. Wolfram. *A New Kind of Science*. Wolfram Media, Inc., 2002. ISBN 1-57955-008-8.

[74] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Computing Surveys*, 38(4), 2006.