DOI: 10.1111/cgf.142664
EUROGRAPHICS 2021
H. Rushmeier and K. Bühler
(Guest Editors)

Volume 40 (2021), Number 2
STAR – State of The Art Report

Algorithms for Microscopic Crowd Simulation: Advancements in the 2010s

W. van Toll[1] and J. Pettré[1]

[1]Univ Rennes, Inria, CNRS, IRISA, France

Abstract

The real-time simulation of human crowds has many applications. Simulating how the people in a crowd move through an environment is an active and ever-growing research topic. Most research focuses on microscopic (or 'agent-based') crowd-simulation methods that model the behavior of each individual person, from which collective behavior can then emerge.
This state-of-the-art report analyzes how the research on microscopic crowd simulation has advanced since the year 2010. We focus on the most popular research area within the microscopic paradigm, which is local navigation, and most notably collision avoidance between agents. We discuss the four most popular categories of algorithms in this area (force-based, velocity-based, vision-based, and data-driven) that have either emerged or grown in the last decade. We also analyze the conceptual and computational (dis)advantages of each category. Next, we extend the discussion to other types of behavior or navigation (such as group behavior and the combination with path planning), and we review work on evaluating the quality of simulations.
Based on the observed advancements in the 2010s, we conclude by predicting how the research area of microscopic crowd simulation will evolve in the future. Overall, we expect a significant growth in the area of data-driven and learning-based agent navigation, and we expect an increasing number of methods that re-group multiple 'levels' of behavior into one principle. Furthermore, we observe a clear need for new ways to analyze (real or simulated) crowd behavior, which is important for quantifying the realism of a simulation and for choosing the right algorithms at the right time.

CCS Concepts

• ***Computing methodologies*** → ***Motion path planning; Real-time simulation; Intelligent agents;***

1. Introduction

A human crowd is a gathering of many humans in a particular place. Crowds are studied in several research disciplines, from social to natural sciences. Researchers are particularly interested in how the humans in a crowd move through space over time; we will refer to this as the *behavior* of a crowd. Crowds exhibit interesting kinds of *collective* behavior that emerges from the behavior of (and interactions between) individual people.

The research area of *crowd simulation* concerns the design and use of simulation algorithms to understand, predict, or reproduce the behavior of human crowds. Realistic computer simulations of human crowds have many applications, ranging from entertainment (such as computer games and movies) to safety and security (such as crowd management and evacuation studies). This research lies at the intersection of many domains, such as computer science, graphics, robotics, physics, cognitive science, traffic theory, civil engineering, and mathematics. The research output can be roughly subdivided into at least three categories: *experiments* for understanding human behavior, *algorithms* for simulating this behavior, and *applications* of these algorithms for specific purposes.

This survey reviews the research developments on *crowd simulation algorithms* since the year 2010. Many of these algorithms have been presented in the communities of computer graphics and robotics. We focus on the class of algorithms known as *microscopic* or *agent-based*, which has received a particular amount of research attention. In a microscopic crowd simulation, each person is modelled as an individual unit (called an *agent*) with its own properties (such as size and walking speed) and motivations (such as a goal position to reach). Thus, microscopic crowd modeling follows a bottom-up approach where the behavior of the crowd follows from the behavior of individuals. To this end, in each simulation step, every agent updates its velocity (i.e. its speed and direction of motion) based on the neighboring agents and obstacles that it observes, and based on certain rules that dictate its local behavior. We will give a more precise outline of this simulation approach in Section 2.

By contrast, a *macroscopic* algorithm models the crowd as one matter and not as a set of individuals. In this paradigm, the crowd's motion is a vector field where each position prescribes a velocity, determined by physical principles such as the conservation of mass. The motion of a person then depends purely on its position in this field. From a modeling perspective, microscopic and macroscopic methods respectively use a Lagrangian and Eulerian description of

the crowd. In traffic theory, the term 'macroscopic' can refer to even more abstract models that do not simulate the traffic itself, but only the relations between statistics such as density and flow. Macroscopic methods are ideal for dense crowds where the differences between individual people are not noticeable. Microscopic methods are preferred whenever these differences *are* important.

In microscopic crowd simulation, it is common to distinguish between different 'levels' of navigation, such as *global* path planning (where an agent plans a 'rough' path to its goal that avoids obstacles in the environment) and *local* behavior (where an agent follows its path while adhering to certain local rules). The latter is also often referred to as 'steering'. By far, the most frequently studied category of local behavior is *collision avoidance*: ensuring that an agent avoids collisions with other (moving) agents. Other examples of local behavior include the behavior of small social groups and the waiting and following behavior in a queue.

The terminology used in this article follows the conventions from computer-graphics literature. Unfortunately, the term 'microscopic crowd simulation' itself has been used ambiguously for several things: the combination of local behavior and global path planning, or local behavior only, or even collision avoidance only. This survey will use the broadest definition, where 'microscopic' concerns the behavior of individual agents at all levels. We will discuss many aspects of microscopic simulation, but with an emphasis on the most active research subtopics: local behavior and collision avoidance.

1.1. Motivation and positioning

Crowd simulation continues to grow as a research topic. Many algorithms are presented each year, especially for collision avoidance and other local behavior, and sometimes there are only small conceptual differences between these algorithms. We therefore see an increasing need for a state-of-the-art review that categorizes recent work and analyzes the (dis)advantages of each category. This can help identify the remaining challenges in this research area, as well as the most promising directions for overcoming them.

We believe that it is particularly interesting to present a survey that focuses on the last decade. Since 2010, a new category of collision-avoidance algorithms has arrived that simulates human vision, and another category of algorithms (*'velocity-based'*) has matured. There have also been many developments regarding *data-driven* methods, partly related to the rise of deep learning.

Several books and surveys give an overview of crowd simulation in its entirety [KPAB15, PAKB16, TM13, XJJD14, YLG*20]. These sources usually also include other subtopics such as global path planning, 3D body animation, real-time rendering, and extracting crowd behavior from video data. As such, they offer a useful broad overview for newcomers to the field, but they are not necessarily meant to give a critical analysis of specific subtopics. In comparison, we will zoom in on microscopic crowd simulation (and local navigation in particular), and we critically discuss its developments in last decade. This will yield more insight into the differences and similarities between recent algorithms, and into the most important future challenges and research directions. There are similar surveys for some of the other subtopics mentioned earlier, including crowd rendering [BPA16] and video analysis [GF17, LCW*15].

An extensive and valuable survey of pedestrian navigation models was presented by Duives et al. in 2013 [DDH13] from the perspective of traffic theory. However, their survey eventually compared methods for different purposes in one table, 'ranking' algorithms by their capability to perform tasks for which they were not always intended. For example, the survey could give a collision-avoidance algorithm the annotation that it does not support global planning, except if its corresponding publication happened to include a separate global-planning algorithm in its experiments. Therefore, the survey strongly focused on whether each publication offers a complete implementation of a crowd-simulation system. By contrast, our state-of-the-art review will deliberately focus on local navigation within the microscopic paradigm, with a deeper discussion of the conceptual differences between the algorithms in this area. We will also assess the general (dis)advantages of *categories* of algorithms, but we do not aim to conclude which specific publication provides the best off-the-shelf solution. Instead, our main goal is to objectively evaluate how the research area has evolved, and to estimate how this evolution will continue.

1.2. Bibliographic data

As explained earlier, crowd simulation contains aspects of many research domains. However, new *algorithms* for crowd simulation are often presented in the area of *computer graphics*, with robotics in second place. Other areas tend to see more literature on the analysis of crowd behavior, or on applications and case studies. Since this survey concerns the algorithmic side, many of our references are from the computer-graphics community, and we consider it to be the most logical community for publication of this survey.

To estimate the growth in research activity regarding this topic, we have performed Google Scholar searches over the keywords 'crowd simulation' limited to each individual year since 2000. Figure 1 shows a growth from very few articles (e.g., 23 in the year 2000) to hundreds per year, with a maximum of 699 results in 2019. (At the time of writing, it is possible that not all publications from the year 2020 can be found yet.) Furthermore, Figure 2 shows the yearly number of publications that contain specific keywords in their title or abstract. For all keywords, the past decade has seen the most publications. Although these numbers are by no means a completely accurate representation of research activity, they clearly illustrate that the topic of crowd simulation has gained significant attention over time.

The literature used for this survey is based on these keyword searches and on the other surveys mentioned in Section 1.1. From this collection of literature, we have selected the papers that introduce new concepts or algorithms, thus filtering out the literature that applies existing algorithms to case studies. Among the remaining papers, we will focus on the most influential ones from the 2010s, meaning the work that has been frequently cited or that is fundamental for a particular category of methods. We will also refer to earlier work if this is useful for the overall discussion.

1.3. Outline

The structure of this article is based on a categorization of the studied literature. We devote multiple sections to *local navigation*, with

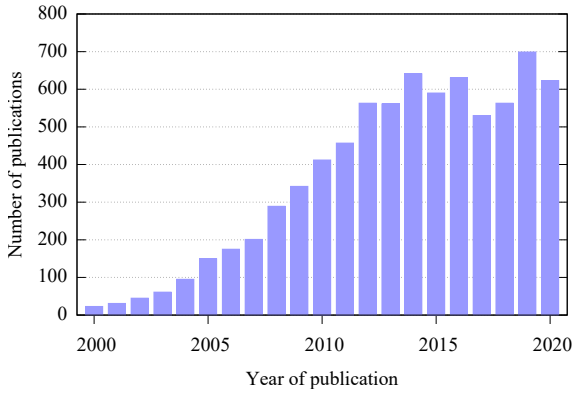


Figure 1: *Number of scientific publications per year that mention the term "crowd simulation", according to a Google Scholar search on January 4, 2021.*

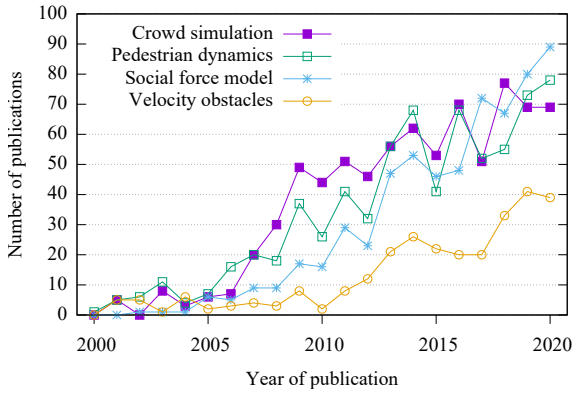


Figure 2: *Number of scientific publications per year that contain certain domain-specific terms in their title or abstract, according to a* `dimensions.ai` *search on January 4, 2021.*

a strong emphasis on *collision avoidance between agents*. This is the most active subtopic of research, and its algorithms can clearly be subdivided into categories. Next to this, we treat other subtopics of the microscopic paradigm in separate sections. Overall, the remainder of this article is structured as follows:

- Section 2 provides definitions of concepts that are commonly used in this research area. This facilitates the discussions in the rest of this article.
- Sections 3–5 describe the main categories of collision-avoidance algorithms from the 2010s: force-based, velocity-based, and vision-based. Each section starts by introducing the category and then zooms in on specific publications from the last decade.
- Section 6 discusses data-driven methods for local behavior. These methods do not necessarily focus on collision avoidance, but on replicating any behavior that occurs in input data.
- Section 7 briefly treats two other types of local behavior that are frequently studied: grouping and following behavior.
- Section 8 extends the discussion to navigation algorithms that play a different role in the overall crowd simulation, such as path following and the combination with global path planning.
- Section 9 gives an overview of the frameworks and implementations that have appeared in the research community.
- Section 10 treats topics related to evaluating the quality or realism of a crowd simulation.
- Finally, Section 11 summarizes this survey, and Section 12 gives an outlook of future research on microscopic crowd simulation.

2. Definitions and preliminaries

This section introduces the terminology and symbols that we will use throughout this article, as well as certain general concepts that are useful to summarize beforehand.

2.1. Domain: Environment, obstacles, agents

In most research on microscopic crowd simulation, the environment in which the crowd moves is simplified to a plane with polygonal obstacles, and the people in the crowd are simplified to disk-shaped particles. All coordinates are specified in meters. Formally, there is a set of m obstacles $OBS = \{O_i\}_{i=0}^{m-1}$ where each obstacle O_i is a simple 2D polygon, and different obstacles do not overlap. Furthermore, there is a set of n agents $AG = \{A_j\}_{j=0}^{n-1}$ where each agent A_j is usually represented by a disk with radius r_j. Each agent also has a preferred walking speed $s_{\text{pref},j}$ (in meters per second) that is typically constant during the simulation.

Let $\mathbf{p}_j$ and $\mathbf{v}_j$ respectively denote the current position and velocity of agent A_j. In every step of the simulation, the task of local navigation is update each $\mathbf{v}_j$ so that agents can progress while avoiding collisions. An agent always has a preferred velocity $\mathbf{v}_{\text{pref}}$ that it would like to attain, e.g. because this velocity leads to a goal position or because it follows a precomputed path. Collision-avoidance algorithms assume that these preferred velocities are given.

In this article, whenever we describe how an algorithm works for a given agent, we will omit the subscript j from the properties mentioned earlier. We will only use these subscripts when it is important to distinguish between different agents.

We acknowledge that not all methods use this particle approximation of agents. Many vision-based and data-driven algorithms (see Sections 5 and 6) do not explicitly rely on it. There is also work that extends the agent representation to capsules [SMTTvdS16], or to a lower-body representation where motion follows from footsteps [SKRF11], or even to a full-body humanoid that can respond to detailed physical interactions [HBM*20]. We will get back to some of these detailed models in Section 8.2. In general, though, the simplifications outlined here are common in our domain.

2.2. Simulation overview

Before we zoom in on the differences between local navigation algorithms, it is essential to understand their position in the overall crowd simulation. In this section, we provide an overview of a typical microscopic crowd simulation, and we make clear on which aspect our survey will focus.

A microscopic crowd simulation consists of discrete timesteps or *frames*, where each frame typically simulates 0.1 seconds. In every frame, each agent A_j performs (at least) the following steps:

1. (Neighbor search) Find the neighboring agents and obstacles within in a certain range. This range is usually a *disk* with a predefined radius, or a disk *section* that only considers neighbors within a certain (viewing) angle. In this survey, we will treat this step as being completely detached from the collision-avoidance algorithm itself. Therefore, we will not compare navigation algorithms by how they choose neighbors, but purely on what they *do* with this information.
2. (Preferred velocity) Based purely on the environment's geometry, compute a *preferred velocity* $\mathbf{v}_{\text{pref}}$ that would send A_j towards its goal if there were no other agents. This can be a velocity that simply points straight to the goal position, or a velocity that lets A_j proceed along a (previously computed) global path around obstacles. In the latter case, this step is also referred to as *path following*.
3. (Local navigation) Based on the neighbor information and on $\mathbf{v}_{\text{pref}}$, compute a new velocity $\mathbf{v}_{\text{new}}$ (or an acceleration vector $\mathbf{a}$) that satisfies certain local criteria. The minimum demand is that the agent's velocity should be close to $\mathbf{v}_{\text{pref}}$ while avoiding collisions with neighbors. However, other criteria can be added as well, such as the desire to stay in a social group.
4. (Movement) Update the agent's position based on its new acceleration or velocity. Most simulations do this via forward Euler integration.

It is common practice to perform one step for all agents before proceeding with the next step. This way, all agents use the same information (i.e. it does not matter in which order they are processed), and each step can easily be *parallellized* for optimal performance. Many modern implementations (see Section 9) use this principle.

In step 2, the preferred velocity $\mathbf{v}_{\text{pref}}$ depends on whether or not the system includes global path planning. Either way, the usual assumption in step 3 is that $\mathbf{v}_{\text{pref}}$ is a good indication of how A_j should navigate amidst the environment's obstacles. Thus, local navigation is mostly concerned with responding to neighboring *agents*. Obstacles are still included for completeness, but the input velocity $\mathbf{v}_{\text{pref}}$ is already assumed to take them into account to some extent.

This survey focuses on the minimal version of step 3: letting each agent A_j find a new velocity (or acceleration) that is close to $\mathbf{v}_{\text{pref}}$ while avoiding collisions with neighboring agents and obstacles. Sections 3–6 revolve fully around this topic. We will mostly discuss collision avoidance *between agents*; for almost all algorithms, the avoidance of *obstacles* uses very similar equations with small adaptations. Also, many collision-avoidance algorithms can be adapted or extended to model other kinds of behavior, such as grouping or following behavior. We will briefly discuss such additional topics in Section 7. After that, Section 8 will extend the discussion to other aspects of the simulation loop.

At this point, we should note that several recent papers explore alternatives to the simulation loop presented here. Weiss et al. [WJLT17] have simulated crowds using position-based dynamics, which directly works with the positions of agents without using the concept of velocities. Karamouzas et al. [KSNG17] have presented a different velocity-optimization scheme based on energy functions, which allows for stable simulations even at very large timesteps. These types of developments are very interesting as they simulate crowds in creative new ways. On the other hand, this conceptual difference makes them difficult to combine with *other* research that still has the classical simulation loop in mind. Future work will have to point out how popular these alternative simulation techniques will become. The remainder of this survey will be based on the traditional simulation loop, also because this leads to the clearest categorization of literature.

2.3. Collision-prediction concepts

Many modern collision-avoidance algorithms use similar principles for *predicting future collisions* between agents. It is useful to list these principles here before discussing each algorithm individually. In recent work, van Toll et al. [vTGL*20] have listed the following four concepts, which are also visualized in Figure 3:

Time to collision $\boldsymbol{ttc}(\mathbf{v}',A_j,A_k)$ The time (in seconds) after which an agent A_j will collide with another agent A_k, assuming that A_j uses $\mathbf{v}'$ and A_k keeps using its current (observed) velocity $\mathbf{v}_k$. The time to collision is infinite if no collision will occur.

Distance to collision $\boldsymbol{dc}(\mathbf{v}',A_j,A_k)$ The distance to the point where A_j and A_k will collide under the same assumptions, i.e. $ttc(\mathbf{v}',A_j,A_k)\cdot||\mathbf{v}'||$. It is infinite if no collision will occur.

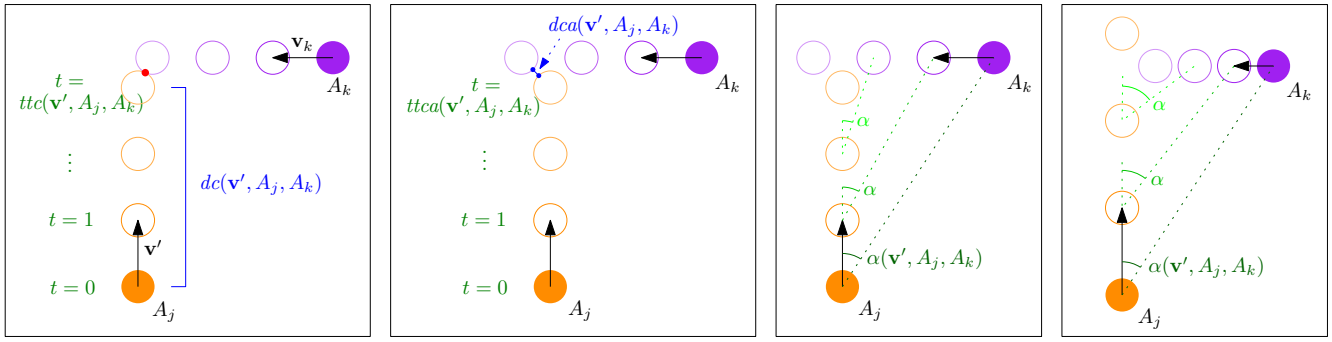
Time to closest approach $\boldsymbol{ttca}(\mathbf{v}',A_j,A_k)$ The time at which the distance between A_j and A_k will be smallest. This is equal to *ttc* if the agents are expected to collide. Unlike the time to collision, it is never infinite, and it can also be negative, namely if the agents are moving away from each other.

Distance of closest approach $\boldsymbol{dca}(\mathbf{v}',A_j,A_k)$ The predicted smallest distance between A_j and A_k, i.e. their distance after $ttca(\mathbf{v}',A_j,A_k)$ seconds. This is zero if the agents will collide.

These concepts can all be computed analytically by solving quadratic equations. Each concept can also be defined for a neighboring *obstacle* instead of a neighboring agent. Also, we can define versions that take the *minimum* time or distance among *all* neighboring agents and obstacles. We will denote these by $TTC(\mathbf{v}',A_j)$, $DC(\mathbf{v}',A_j)$, $TTCA(\mathbf{v}',A_j)$, and $DCA(\mathbf{v}',A_j)$. For example, $TTC(\mathbf{v}',A_j)$ is the time to A_j's first predicted collision with any object if A_j uses velocity $\mathbf{v}'$.

Certain vision-based collision-avoidance algorithms (Section 5) use an alternative concept that is worth mentioning here. The *bearing angle* $\alpha(\mathbf{v}',A_j,A_k)$ is the current angle between the vectors $\mathbf{v}'$ and $\mathbf{p}_k-\mathbf{p}_j$. Its time derivative $\frac{d}{dt}\alpha(\mathbf{v}',A_j,A_k)$ is the amount by which α will currently start changing per second, again assuming that A_j and A_k will respectively keep using $\mathbf{v}'$ and $\mathbf{v}_k$. If the agents are going to collide, this gradient is (close to) zero. Otherwise, it is positive or negative; Figures 3(c) and 3(d) show two examples. Compared to the concepts mentioned earlier, the bearing angle corresponds more closely to what a human pedestrian *sees* (namely another pedestrian moving through its field of view). It can be applied more easily to vision-based methods (Section 5) without requiring perfect information about the positions and velocities of all agents.

These concepts all assume that the velocities of agents remain constant for some time, i.e. they assume that each agent will follow a linear trajectory in the near future. This is a generally accepted assumption. Its simplicity is balanced out by the fact that all agents compute a new velocity in every frame of the simulation; in other words, their collision predictions are re-considered several times



(a) Time to collision (b) Time to closest approach (c) Bearing angle (decreasing) (d) Bearing angle (increasing)

Figure 3: *Overview of collision-prediction concepts between two agents A_j (in orange) and A_k (in purple). $\mathbf{v}'$ is a hypothetical velocity for A_j, and $\mathbf{v}_k$ is the current observed velocity of A_k. (a) The time and distance to collision. In this example, the agents collide. (b) The time to and distance of closest approach. In this example, the agents do not collide. (c) A bearing angle that decreases over time. (d) A bearing angle that increases over time.*

per second. Still, some algorithms exchange this linear-trajectory prediction with something more advanced, as we will see in Section 8. Also, several data-driven methods (Section 6) base their prediction on input data without making an explicit assumption about linearity. However, most 'classical' collision-avoidance algorithms use the prediction concepts summarized here.

3. Force-based collision avoidance

In 1995, the *social-force model* (SFM) by Helbing and Molnár [HM95] was the first model specifically designed for collision avoidance in crowds. This model treats each agent A_j as a particle that experiences two main types of forces: an *attractive* force $\mathbf{F}_{\text{att}}$ towards the goal position, and *repulsive* forces from obstacles and other agents. For neighboring agents, A_j experiences a separate force $\mathbf{F}_{jk}$ per neighbor A_k, and this force depends on the positions and velocities of A_j and A_k in a certain way. Thus, although the original SFM does not yet explicitly use the concept of 'time to collision', it already bears similarities to it.

The term 'social force model' has been almost synonymous with 'collision avoidance' for a long time, at least until the introduction of velocity-based algorithms in the late 2000s (which we will discuss in Section 4). Even today, the SFM is still widely used. The concept is easy to implement, it works sufficiently well for many applications, and it is intuitive to extend with extra agent behaviors simply by introducing new forces. In fact, based on in Figure 2, the term 'social force model' has been roughly equally popular as 'crowd simulation' when it comes to its usage in titles and abstracts. Its popularity has even grown relatively fast in recent years, despite the introduction of other (more advanced) algorithms for local behavior. However, most of these recent publications apply the standard social-force model to a particular case study, or they add new forces to simulate particular effects, such as groups, panic propagation, waiting behavior, lateral motion, or pushing behavior.

In this section, we will review the evolution of force-based collision avoidance throughout the past decade. As mentioned before in Section 2.2, we focus on collision avoidance *among agents* to explain the differences between methods. All methods use a very similar force to let agents avoid static *obstacles*. Note that we will only treat articles that apply fundamental changes to the concept of collision avoidance. This excludes the aforementioned work on other local behaviors (some of which we will briefly treat in Section 7) and any case studies that use the standard SFM.

The most important conceptual improvement upon the standard social-force model has been the introduction of *collision prediction*. In these models, the force $\mathbf{F}_{jk}$ that an agent A_j receives from a neighboring agent A_k does not simply depend on the agents' current velocities and positions, but also on whether these actually lead to a *collision* in the future. We identify two noteworthy publications from the 2010s based on this concept, which we will discuss in the next two subsections.

Although technically not within our decade of focus, the 'predictive collision-avoidance model' by Karamouzas et al. from 2009 [KHvBO09] is worth mentioning here, as it was (to our knowledge) the first force-based method to use the concept of time to collision. In their work, the force $\mathbf{F}_{jk}$ is a simple function of the predicted time to collision between A_j and A_k, i.e. of $ttc(\mathbf{v}_j, A_j, A_k)$. The *magnitude* of $\mathbf{F}_{jk}$ depends on *when* this collision will occur, and the *direction* of $\mathbf{F}_{jk}$ depends on *how* the agents will be positioned with respect to each other at that time. If no collision is expected ($ttc(\mathbf{v}_j, A_j, A_k) = \infty$), then $\mathbf{F}_{jk} = \mathbf{0}$.

3.1. SFM with time to collision (Zanlungo et al. 2011)

In 2011, Zanlungo et al. [ZIK11] compared the equations of the most common SFM variants that existed up until then. They observed that several models used a (constant) 'time window' parameter τ indicating how far an agent A_j looks into the future to compute its forces. The force $\mathbf{F}_{jk}$ (exerted on A_j by a neighboring agent A_k) is then based on the predicted positions of A_j and A_k after τ seconds. Zanlungo et al. replaced this parameter by the *time to collision*. More specifically, they used $TTC(\mathbf{v}, A_j)$, i.e. the time to the first predicted collision if A_j keeps using its current velocity $\mathbf{v}$.

Consequently, agents automatically adapt their amount of 'lookahead' based on the first problem that they expect to encounter. The authors finally showed that this improved model could be calibrated to match real-world data more closely than previous SFM variants.

This differs from the model by Karamouzas et al. [KHvBO09] in a number of ways. Most importantly, Karamouzas et al. use $\mathbf{F}_{jk} = \mathbf{0}$ if A_j and A_k are not on collision course. In such cases, the force $\mathbf{F}_{jk}$ by Zanlungo et al. is still non-zero, so agents may display avoidance behavior when it is not necessary. Another difference is that Karamouzas et al. [KHvBO09] use the *specific* time to collision $ttc(\mathbf{v}, A_j, A_k)$ to scale the force *per neighbor* A_k, whereas Zanlungo et al. [ZIK11] use the *overall* time to collision $TTC(\mathbf{v}, A_j)$ to scale the forces of *all* neighbors. Overall, the approach by Karamouzas et al. [KHvBO09] seems more intuitive: it treats the avoidance of each neighbor as an independent task, and it ignores neighbors for which no problems are expected. On the other hand, the model of Zanlungo et al. [ZIK11] is a direct extension of the SFM and may be preferred in its corresponding community.

Unfortunately, to our knowledge, a direct comparison between these models has not been performed. In literature, a new collision-avoidance algorithm is usually compared to the standard SFM [HM95] (a 'baseline' upon which it is easy to improve) or to RVO or ORCA [vdBLM08, vdBGLM11] (two popular velocity-based methods that do not necessarily aim for 'human-like' results).

3.2. Universal Power Law (Karamouzas et al. 2014)

In 2014, Karamouzas et al. introduced the popular 'Universal Power Law' method [KSG14]. Compared to other force-based models with collision prediction, this paper builds its explanation on a different principle: pedestrians attempt to *minimize the energy* they spend on interactions, knowing that a predicted collision has a lower probability of actually occurring if it is farther away. The authors show that real-world data supports this theory to some extent.

This argumentation results in an equation for $\mathbf{F}_{jk}$ that conceptually fits between the SFM extension of Zanlungo et al. [ZIK11] and the previous predictive force by Karamouzas et al. [KHvBO09]. Just like in these two models, $\mathbf{F}_{jk}$ depends on the time to collision between A_j and A_k based on their current velocities. The model is 'Helbing-like' because it is based on energy, and it is 'Karamouzas-like' because $\mathbf{F}_{jk} = \mathbf{0}$ when there is no predicted collision.

This method has become very popular, possibly because it is conceptually simpler (and computationally lighter) than the velocity-based methods that already existed at the time, such as RVO and ORCA. Thus, after years of increasingly complex solutions, this work seemed to bring collision avoidance 'back to basic' with good results. However, in theory, a velocity-based model should be able to handle more complicated scenarios than a force-based model, as we will explain in Section 4.

3.3. Summary

The force-based models discussed in this section explicitly use the concept of 'time to collision' for the force $\mathbf{F}_{jk}$ that an agent A_j receives from a neighbor A_k. The exact equation for $\mathbf{F}_{jk}$ differs per method: it is either a direct extensions of the social-force model [ZIK11] or a new equation that ignores A_k if it is not on collision course with A_j [KHvBO09, KSG14].

Compared to the regular SFM, these models are more adaptive to what an agent perceives. They also remove a 'time window' parameter from the system. Having to compute the time to collision does increase the method's computational overhead. However, force-based methods remain faster than velocity-based methods (Section 4), which usually involve more of these computations.

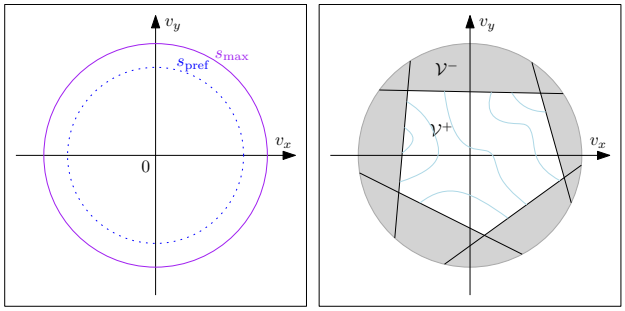
4. Velocity-based collision avoidance

Since 2007, the field of crowd simulation has seen the development of so-called *velocity-based* algorithms for local navigation. Compared to force-based models with collision prediction (Section 3), velocity-based models take the concept of prediction even further. Their general principle is to let each agent *actively choose* its next velocity (i.e. a speed and direction) by considering many possible velocities, evaluating each option according to certain criteria (including whether they successfully avoid collisions), and finally choosing the best option.

This principle fundamentally changes how an agent moves in interaction with its neighbors. By actively considering multiple velocities and by predicting what their consequences will be, an agent can adjust its trajectory in an anticipated way. By contrast, force-based methods only update an agent based on how it is currently moving; they do not actively consider what would happen if this agent did something else. One consequence of this difference is that velocity-based methods are computationally heavier than force-based methods. However, they also tend to yield better results, and they are considered to better reflect human behavior.

On a more detailed level, two main types of velocity-based methods can be distinguished. Let the *velocity space* $\mathcal{V}$ be the (continuous) set of possible velocities that an agent can use, as shown in Figure 4(a). The first type of velocity-based method defines a *cost function* $\mathcal{C}$ that assigns a scalar cost to each velocity in $\mathcal{V}$, where a lower cost indicates that a velocity is a 'more attractive' option to choose. Methods of this type differ in how they define the cost function $\mathcal{C}$ itself, but also in how they search through $\mathcal{V}$ to find a good velocity to use. The second type of velocity-based method decomposes $\mathcal{V}$ into two parts: a set of *admissible* velocities $\mathcal{V}^+$ that allow a collision-free path within a certain time window, and a set of *inadmissible* velocities $\mathcal{V}^-$ that will lead to a collision in the near future. These methods then look for the best velocity inside $\mathcal{V}^+$. This final step is again done by optimizing over a cost function $\mathcal{C}$. Figure 4(b) summarizes this idea.

Conceptually, the line between these two types of algorithms is blurry. After all, having two subspaces $\mathcal{V}^-$ and $\mathcal{V}^+$ is technically equivalent to having a cost function $\mathcal{C}$ that assigns an infinite cost to inadmissible velocities. The largest remaining difference is that methods based on an explicit $\mathcal{V}^-$ have an *analytical solution* for the optimal velocity. Other methods do not have this: they simply try out many 'sample' velocities $\mathbf{v}'$ and choose the sample with the lowest cost. This latter approach is computationally heavier. On the other hand, it also offers more flexibility for the cost function $\mathcal{C}$, which can then have any form. By contrast, an analytical solution relies on very specific choices for $\mathcal{V}^-$ and $\mathcal{C}$.



(a) Velocity space (b) Decomposition + Cost function

Figure 4: *(a) The velocity space $\mathcal{V}$ is the set of all velocities that an agent A_j can use. It can be visualized as a disk with radius s_{max} (the agent's maximum walking speed). (b) Velocity-based collision avoidance defines a cost function over $\mathcal{V}$ (shown here via blue isolines), or possibly over a subset $\mathcal{V}^+$ of admissible velocities.*

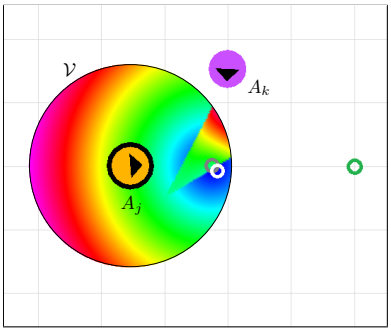


Figure 5: *Example of the RVO collision-avoidance algorithm. An agent A_j (in orange) wants to walk to the green circle on the right. The cost function C for the entire velocity space $\mathcal{V}$ is drawn around A_j, at a scale from blue (low cost) to pink (high cost). The agent's preferred velocity is shown as a gray circle. However, a neighboring agent A_k (in purple) induces a cone-shaped range of velocities for A_j that will lead to a collision in the future. To avoid this collision, A_j chooses a slightly different velocity, shown in white. RVO finds this velocity by evaluating random samples in $\mathcal{V}$.*

In 2007, Paris et al. [PPD07] were the first to present a velocity-based collision-avoidance algorithm. This model defines the inadmissible velocities induced by each neighboring agent, and it uses a cost function to choose among the remaining admissible velocities. Due to a somewhat cumbersome formulation and due to missing details, the method has never become commonly used, but it remains the first model that can be categorized as velocity-based.

In 2008, the RVO (Reciprocal Velocity Obstacles) method of van den Berg et al. [vdBLM08] truly popularized this category. This work formulates $\mathcal{V}^-$ as a union of *velocity obstacles* (VOs), where each VO has an explicit cone-like shape. The 'reciprocal' aspect of the method lies in the assumption that any two agents will always spend an equal amount of effort on avoiding each other. This makes the behavior of agents smoother and less indecisive. Interestingly, the original *implementation* of the RVO algorithm uses a cost function and sampling, without making use of the analytical form of VOs. Still, the article contains mathematical foundations that have frequently been re-used and enhanced since then. The RVO method therefore continues to be highly influential to this day. Figure 5 shows a simple example of RVO in practice, with only one neighboring agent (and thus only one velocity obstacle).

Velocity-based approaches have continued to develop in the 2010, but they are all based on the core principles that we have described here. This section reviews the developments in this category. We will first discuss methods that use cost functions and sampling, followed by methods with an analytical solution.

4.1. Methods with a cost function and sampling

The original RVO implementation [vdBLM08] was the first example of sampling a cost function in velocity space. In the early 2010s, two other methods based on this principle have been developed.

4.1.1. First method (Karamouzas and Overmars 2010)

In 2010, Karamouzas and Overmars introduced the term 'velocity-based approach' in the title of their article [KO10]. They described the concept of evaluating many 'candidate' velocities for an agent and computing a cost for each candidate. Although this concept was used before in the implementation of RVO [vdBLM08], this paper was the first to revolve fully around it. In this method, agents try out various speeds and directions via regular sampling. The range of allowed speeds and directions depends on the time to collision for the agent's *preferred* velocity, i.e. on $TTC(\mathbf{v}_{\text{pref}}, A_j)$. In particular, if $\mathbf{v}_{\text{pref}}$ does not lead to a predicted collision, then this velocity will always be chosen. Limiting these ranges mostly improves the method's computational efficiency.

For each sample velocity $\mathbf{v}'$ in the allowed range, the cost depends on three factors: the difference between $\mathbf{v}'$ and the current velocity, the deviation from the preferred velocity $\mathbf{v}_{\text{pref}}$, and the time to collision $TTC(\mathbf{v}', A_j)$. Each component of this cost function has its own weight that can be tuned.

A conceptual strength of this algorithm is that it is based on real-world data. The authors have conducted experiments in which two humans were instructed to walk forward while avoiding collisions. The results of these experiments have led to specific equations and parameters in the algorithm.

In line with this, the authors argue that their algorithm yields smoother and more human-like behavior than RVO [vdBLM08], which treats collision avoidance purely mathematically. They also claim that agents respond to collisions sooner than RVO, which would imply that agents do not need to stand still to avoid collisions at the last moment. However, this comparison is theoretically questionable if we look at the details. Both algorithms rate a candidate velocity $\mathbf{v}'$ based on its time to collision $TTC(\mathbf{v}')$ (among other factors), and both algorithms technically allow agents to stand still.

4.1.2. Simplified variant (Moussaid et al. 2011)

A year later, Moussaid et al. [MHT11] presented a similar method with a few conceptual differences. First, the ranges of speeds and

directions that an agent can use remain constant; these do not depend on the agent's current situation. Second, their cost function only depends on the time to collision and the angular deviation from the preferred velocity. There is no 'energy term' that uses the difference with the agent's *current* velocity. Third, whenever an agent does collide with another agent or an obstacle, the method applies a *contact force* using existing equations by Helbing et al. [HFV00]. By contrast, Karamouzas and Overmars [KO10] used a special cost function for agents in a colliding state. Forces are an easier way to ensure that collisions get resolved.

Finally, Moussaid et al. assume that a person always walks at its preferred speed s_{pref} unless a collision is bound to happen soon. In their method, an agent first finds an optimal *direction* of motion (via regular sampling), assuming that it will walk at its preferred speed. Given this optimal direction, it then computes a matching *speed*, which is usually s_{pref} except if this speed would lead to collision. Consequently, agents prefer directional changes over speed changes, and they walk at full speed as much as possible.

The method by Moussaid et al. is conceptually simpler than the one by Karamouzas and Overmars [KO10], with fewer parameters to tune and without adaptive sampling ranges. It can also be computationally more efficient because it only samples a 1D range of directions, and not a 2D range of speeds and directions combined. However, this difference in performance is scenario-dependent as as the sampling range of Karamouzas and Overmars is adaptive. Finally, it is not clear which method yields more realistic *behavior*. While Moussaid et al. do compare their simulations against empirical data, the method itself is not based on real-world observations.

4.2. Methods with an analytical solution

The research group behind Reciprocal Velocity Obstacles (RVO) has created several variants or successors of their algorithm in the 2010s. Some of these are logical extensions of RVO for particular purposes, such as the inclusion of acceleration constraints for robots [vdBSGM11], the inclusion of psychological factors to simulate e.g. escape panic [GKLM11], or modelling agents as *ellipses* (instead of disks) to handle high-density scenarios [NBM17]. Similarly to extensions of the *social-force model* for these purposes, we will not treat such work in this survey.

Instead, we will focus on two *other* RVO-inspired methods that treat the 'standard' collision-avoidance problem in fundamentally new ways. We believe that these methods are the most important ones for describing the evolution of this category of algorithms.

4.2.1. ORCA (van den Berg et al. 2009–2011)

The main successor of RVO, named *Optimal Reciprocal Collision Avoidance (ORCA)*, was presented at a conference in 2009, but its official corresponding publication is from 2011 [vdBGLM11]. Compared to RVO, ORCA transforms the mathematical definition of the collision-avoidance problem, so that an agent can compute its optimal velocity *analytically*, i.e. without the need for sampling. Recall that RVO was effectively a sampling-based algorithm, despite its underlying mathematical concepts.

In the original RVO formulation, each neighbor A_k of an agent A_j induces a velocity obstacle with a truncated cone shape, representing a set of 'forbidden' velocities that would lead to a collision in the near future. In ORCA, each neighbor instead induces a *line* that separates the velocity space $\mathcal{V}$ into two half-planes, containing 'optimal' and 'non-optimal' velocities. The 'optimal' velocities avoid a collision with A_k while staying close to a given optimization velocity $\mathbf{v}_j^{\text{opt}}$, which is typically equal to the current velocity $\mathbf{v}_j$. An ORCA line can always be obtained analytically from an RVO cone. The ORCA lines of all neighbors combined induce an intersection of half-planes, constituting the admissible velocity space $\mathcal{V}^+$. If this intersection I is not empty, then A_j should find the velocity inside I that is closest to its preferred velocity $\mathbf{v}_{\text{pref}}$. If I is empty (which can happen in densely packed conditions), A_j should find the 'least forbidden' velocity according to a geometric definition. In both cases, the answer can be computed via linear programming, because the velocity constraints are now lines instead of more complex shapes.

Like other velocity-based methods, ORCA optimizes a cost function in velocity space, but this time the optimization has an analytical solution. This makes ORCA computationally more efficient than RVO and other sampling-based methods. In fact, the authors have released the source code of ORCA under the name 'RVO2', suggesting that they consider ORCA to be a direct successor and improvement of RVO. On the other hand, ORCA's solution exists thanks to a very specific geometric interpretation of the collision-avoidance problem. It is relatively difficult to extend this interpretation to other types of behavior. By contrast, a sampling-based solution can theoretically use any cost function, which makes it easier to combine different behaviors with their own weights or priorities. Thus, ORCA can produce one type of behavior very efficiently, whereas other methods can be adapted more easily in exchange for a higher computational load.

4.2.2. PLEdestrians (Guy et al. 2010)

The PLEdestrians method by Guy et al. [GCC*10] has many similarities to ORCA, but it is presented from a different angle, focusing on obtaining a similarity to human behavior. It combines ORCA concepts with a cost function comparable to those of Section 4.1. For this combination, the optimal velocity for an agent can (again) be computed analytically in a specific way.

The cost function $\mathcal{C}$ of PLEdestrians is based on the concept of energy minimization, i.e. on the 'principle of least effort' (PLE). The authors propose an energy function that captures how much energy (or 'effort') a pedestrian spends over time. An agent A_j attempts to minimize this effort by selecting a suitable velocity in each frame. The effort of a candidate velocity $\mathbf{v}'$ depends on the time to the first collision $TTC(A_j, \mathbf{v}')$ and on the estimated 'detour length' for steering the agent back to its goal later. This is conceptually very similar to the cost function that Moussaid et al. [MHT11] would present one year later.

This method uses the same definition of the admissible velocities $\mathcal{V}^+$ as ORCA, and then it finds the velocity in $\mathcal{V}^+$ that minimizes the mentioned cost function. Guy et al. show that this optimal velocity either is the agent's preferred velocity $\mathbf{v}_{\text{pref}}$ itself, or it lies on the boundary of $\mathcal{V}^+$. In the second case, the solution can be found by solving a quadratic equation for each ORCA line.

The authors show that PLEdestrians yields more energy-efficient

paths than several other methods, including the standard social-force model (from Section 3) and RVO. They also show that, in certain scenarios, the simulated agents match real humans in terms of the fundamental diagram (the average relation between crowd density and walking speed). Implementation details aside, PLEdestrians is essentially ORCA with a different cost function that is potentially more human-like and less 'robotic'. However, the cost function is still quite specific, to ensure that the optimization problem has a closed-form solution. For any other cost function (e.g. to capture other types of behavior), such a closed-form solution may not exist, and sampling may be necessary.

4.3. Summary

Research on velocity-based collision avoidance thrived around the year 2010. Velocity-based methods differ from force-based methods in that they let an agent analyze *all possible velocities* that it can use, instead of anticipating the effects of only *one* velocity. Consequently, a velocity-based algorithm is computationally heavier, but also theoretically capable of handling more difficult scenarios.

Overall, we can distinguish between sampling-based algorithms that can use any cost function [KO10, MHT11], and 'closed-form' algorithms that are faster but that rely on specific mathematical definitions [vdBGLM11, GCC*10]. Next to extensions for modelling specific effects, this category has not seen any substantial improvements *later* in the 2010s. However, the basis has remained highly popular, also thanks to the freely available source code of RVO and ORCA. In terms of research, the focus seems to have shifted to vision-based and data-driven methods, which we will discuss in the next two sections.

5. Vision-based collision avoidance

The category of *vision-based* navigation algorithms is based on the fact that the locomotion of humans is primarily controlled by their sense of vision. These algorithms attempt to replicate how humans navigate literally based on what they *see*, and not on perfect 'global' information about the positions and velocities of all agents.

The term 'vision-based' has also been used in some of the collision-avoidance literature from Sections 3 and 4. For example, Moussaid et al. [MHT11] use this term because agents only consider neighbors inside a certain field of view. However, the algorithms from Sections 3 and 4 are all purely geometric. In this survey, we reserve the term 'vision-based' for methods that actually attempt to replicate visual perception.

Within the vision-based category of methods, we can again identify two subcategories. The first, which we call *visually-driven steering*, defines behavior based on variables close to what people visually perceive, i.e. objects moving through their field of view, without perfect information in world coordinates. This type of methods was founded before 2010 by Warren et al. [War06, FW07, WF08], a group of psychology researchers who combine modeling with real-world experiments. Their work also introduced collision prediction based on the bearing angle, which we summarized earlier in Section 2.3. In the 2010s, this concept itself has not fundamentally changed. Park et al. [PRY13] have presented a collision-avoidance method based on the 'gaze movement angle', which is essentially the same idea using different terminology. Apart from this, Warren et al. have extended visually-driven steering to other behaviors, such as following behavior in unidirectional and bidirectional crowd flows. We will revisit this topic in Section 7.

The second category takes visually-driven steering one step further by equipping the agents with a *virtual retina*: an image onto which visual information is projected, inspired by the human eye. We therefore refer to this as *retina-based* steering. Although this idea was already implemented in 1990 by Renault et al. [RTT90], it took until the 2010s to be applied to crowds.

In more detail, the main principle of *retina-based* approaches is to equip agents with a 'synthetic vision' that is intended to resemble human visual perception. Rather than using the simplified representation of disks and polygons on a plane, each agent uses a *graphical rendering* of its field of view. The behavior of an agent is then based on the pixel information in this 'virtual retina'. In other words, interaction with other agents or objects is abstracted to interaction with a matrix of pixels. The methods of this category differ in what exactly they render onto the virtual retina, and in how they use this information to steer an agent.

Retina-based algorithms are computationally more expensive than velocity-based or force-based algorithms, simply because they do more work per agent. In exchange, they aim to represent human perception more accurately. These methods are usually designed with low to medium crowd densities in mind, for which performance then remains manageable. On a related note, the 'retina-specific' parts of such a method can sometimes be seen separately from the other components. That is, some concepts can also be applied to the simpler 'geometric' domain from Sections 3–5, resulting in higher performance and suitability for large crowds. This is the case for the first two methods that we will discuss next.

5.1. First retina-based method (Ondřej et al. 2010)

To our knowledge, the first retina-based navigation algorithm for crowds was presented by Ondřej et al. [OPOD10]. Just like several force-based and velocity-based methods, this algorithm is based on *collision prediction*, and it is *reactive* in that an agent only changes its velocity when a future collision is predicted. However, there are three important differences to previously discussed methods.

First, and most obviously, an agent A_j now uses information that is projected onto a virtual retina. Neighboring agents are rendered as cones onto the retina, and obstacles are rendered with their original 3D shape. For each non-empty pixel in the corresponding retina image, A_j computes if a collision with the corresponding agent or obstacle will happen. Thus, the collision avoidance is handled 'per pixel' instead of per object.

Second, the collision prediction itself is based on different indicators: the *time to closest approach* and the time derivative of the *bearing angle*; see Section 2.3 for a summary of these concepts. The method uses the bearing-angle derivative instead of (for example) the time to collision because it is directly linked to *changing* the agent's angle of motion.

Third, namely, this algorithm controls an agent's speed and direction separately. From the agent's retina image, the algorithm

determines the pixels that yield the highest risk of collision. The agent's motion is then adjusted to avoid this 'primary threat'. Overall, agents will *turn* to avoid a collision in the near future, and *decelerate* only for imminent collision cases. (The velocity-based method by Moussaid et al. [MHT11] is based on the same idea: agents will primarily change their direction and not their speed.)

Note that the second and third differences can also be applied to a 'retina-less' domain. Although the virtual retina was the most innovative component of this method at the time, the concept of adjusting the direction and speed separately would be re-used by future vision-based methods as well.

5.2. Using gradients (Dutra et al. 2017)

In 2017, Dutra et al. [DMCN*17] continued with the idea of steering an agent based on the information on its virtual retina. However, whereas Ondřej et al. [OPOD10] directly updated an agent's angle and speed according to certain rules, Dutra et al. [DMCN*17] define a *navigation function* $\mathcal{C}$ and steer an agent according to the gradient of $\mathcal{C}$. Just like in the velocity-based methods of Section 4, the domain of $\mathcal{C}$ is the agent's *velocity space* $\mathcal{V}$. In other words, the function (theoretically) assigns a cost to each possible velocity $\mathbf{v}'$ for the agent, based on certain quality criteria such as whether $\mathbf{v}'$ avoids future collisions. However, this time, the cost function is based on the agent's virtual retina, and not on the 'clean' geometric information used by force-based and velocity-based algorithms. Another difference is that this algorithm does not actively explore the entire velocity space. Instead, it computes the *gradient* of $\mathcal{C}$ at the agent's *current velocity*, i.e. $\nabla\mathcal{C}(\mathbf{v})$, and it steers the agent in the opposite direction, so as to 'greedily' move to a lower-cost velocity.

Collision avoidance is based on two variables here: the *time* and the *distance* to closest approach, i.e. $TTCA(\mathbf{v}',A_j)$ and $DCA(\mathbf{v}',A_j)$. The navigation function $\mathcal{C}$ penalizes low perceived values for both variables, so an agent will try to keep its neighbors far away in both space and time. For goal reaching, $\mathcal{C}$ contains a component that penalizes the angle to the preferred velocity $\mathbf{v}_{\text{pref}}$. The gradient of $\mathcal{C}$ is computed along two axes: the agent's *speed* and *direction*. (In both axes, $\nabla\mathcal{C}(\mathbf{v})$ has a closed form that is easy to evaluate at runtime.) Thus, the algorithm effectively controls the speed and direction separately, but based on the same function.

It is interesting to note that the navigation function does not define a precise threshold for the risk of collision. Thus, agents will adjust their separation distances depending on the case at hand. At a low crowd density, even low collision risks (i.e. for which there are already margins of distance between agents) will influence the value of the navigation function, and its gradient will cause the agents to deviate as much as possible. At a higher density, closer obstacles take up more space in the agent's retina, and the algorithm implicitly gives more weight to avoiding them.

Again, the retina-related aspects of this method can be seen separately from the navigation aspects. The function $\mathcal{C}$, as well as the concept of gradient-based steering, can also be applied to the simpler domain of disk-shaped agents in 2D. In 2020, van Toll et al. [vTGL*20] have suggested such a 'retina-less' variant of this navigation algorithm, where the costs per neighboring agent are scaled to mimick the effect of pixel occupancy. This variant is easier to implement and computationally more efficient, while still capturing most of the essence of the original algorithm.

On a more critical note, gradient-based steering is algorithmically less 'intelligent' than exploring the full velocity space. An agent will make only 'greedy' decisions to change its current velocity, without actively considering velocities that are very different. However, it may be that this seemingly imperfect navigation approach is actually closer to human behavior. More experimentation and analysis is required before any conclusions can be drawn.

5.3. Using optical flow and light (Lopez et al. 2019)

So far, we have seen that it is possible to simulate the local navigation behavior of agents by equipping them with a virtual retina. This retina defines how interactions with surrounding obstacles are selected, combined, and weighted. However, the variables associated with its pixels are not always quantities that would be directly perceptible to the human eye. Both Ondřej et al. [OPOD10] and Dutra et al. [DMCN*17] trace each occupied pixel back to the corresponding agent or obstacle, from which they can then extract 'perfect' information like the velocity or the time to closest approach.

In contrast to this, López et al. have presented two navigation methods [LCMP19b,LCMP19a] that aim to get even closer to how the human brain directly processes visual information. These algorithms steer an agent based on the (RGB) *images* that it perceives, as well as on the *dense optical flow* generated by the succession of these images. Optical flow is the apparent movement of pixels in an image, and the term 'dense' means that this apparent motion is estimated for each pixel separately. Lopez et al. first used a *synthetic* optical flow that was still based on knowledge of the underlying objects [LCMP19b]. Later, they used a *digital* optical flow based purely on the differences between images [LCMP19a]. The second article also introduced a way to attract agents towards lighter areas, where this optical flow can be more accurately obtained.

The agents finally navigate by following the gradient of a cost function, similarly to Dutra et al. [DMCN*17]. However, the function itself is now based on more 'raw' visual information. Lopez et al. show how to use this information to achieve various behaviors such as goal reaching, collision avoidance, and following. For example, the risk of collision with an object can be estimated based on that object's *focus of expansion* in the image. Collision avoidance can thus be achieved by steering the agent away from this point.

This approach has interesting potential for *robot navigation* in a crowd. If a robot should make decisions based on what it perceives with a camera, it is useful for the algorithm to be purely based on visual data. On the other hand, it remains to be seen whether this 'lack of interpretation' makes sense for simulating *human* navigation. Although the algorithm is theoretically closer to reality, it may over-complicate the original navigation problem, as it abstracts away useful information. Finally, on a practical level, this method is not yet suitable for large crowds in real-time, due to the computational load of calculating the optical flow for each agent.

5.4. Summary

Vision-based collision avoidance has evolved significantly in the 2010s. The main development was the introduction of *retina-based* algorithms that literally try to simulate how the human eye works. Within this category, research has moved to the use of visual information only, without any reasoning about objects in world coordinates. This development is also useful for the navigation of robots equipped with cameras. For real-time crowd simulation, though, it remains to be seen whether this is the ideal angle of approach.

Compared to velocity-based methods that explore an entire space of possible velocities, vision-based methods use gradient-based steering which seems more limited. It is interesting that some aspects of retina-based algorithms could also work in the 'classical' domain of disk-shape agents in 2D. This brings vision-based and velocity-based algorithms closer together, which will allow for better comparisons between the two categories.

6. Data-driven local behavior

All methods discussed so far model human behavior via a set of concrete 'hand-made' rules or functions. By definition, these rules can only replicate the behavior of an individual person to a certain extent. In contrast to this, *data-driven* methods are directly based on input data, which is usually a set of trajectories obtained from a real human crowd. These methods aim to replicate their input data in a more abstract sense, without explicitly defining the behavioral rules themselves. As such, a data-driven crowd simulation can (in theory) produce more specific or subtle behaviors that are difficult to capture in simple rules. Another motivation is that a data-driven model will automatically behave differently when the input data changes, without requiring explicit knowledge of what these behavioral differences are. Research in this area started before 2010, but it has gained popularity in the last decade, partly thanks to the recent developments in deep learning.

Note that data-driven methods do not focus specifically on collision avoidance, but simply on reproducing any behavior that they receive as input. Still, we can identify different 'levels' of agent behavior in the same way as in other models: for example, data-driven crowd simulation keeps the traditional distinction between global paths and local behavior. In this section, we discuss algorithms that produce local behavior, i.e. an agent's short-term decision-making that is repeated at a certain frequency. We will refer to this collectively as *data-driven local behavior*. This excludes work that attempts to reproduce higher-level motion patterns in the crowd, such as the distribution of paths throughout an environment.

Like in many other research areas, a large amount of recent work on crowd simulation has a data-driven or (deep) learning-based aspect to it. We emphasize again that this section focuses on *data-driven models for local navigation*. Articles that apply data-driven or learning techniques to *other* aspects of crowd research (e.g. user-controlled motion editing, global navigation, parameter calibration, tracking, or the evaluation of results) are out of scope. Data-driven evaluation work will be discussed in Section 10.

The main idea of data-driven local behavior is to base the behavior of each individual agent on input data. Early work following this idea was published in 2007 by Lerner et al. [LCL07] and by Lee et al. [LCHL07]. These two methods both use a database of examples of pedestrian interactions, in which each entry is a short fragment of crowd motion centered around one particular pedestrian (say A_j). An entry describes the state of the surrounding people and obstacles relatively to A_j (such as their relative positions and velocities), as well as the way in which A_j continues moving in response to this. At runtime, each agent in the simulation compares its surroundings to the database, searches for the most similar entry, and uses the behavior stored in that entry. Thus, at a given moment in time, the behavior of each agent is a copy of the behavior in the most similar input record. The two methods from 2007 differ, for instance, in how exactly they search the database for an entry to use.

These early models cannot guarantee that agents will avoid all collisions. In that sense, they are not exactly 'collision-avoidance-like' algorithms to be performed at a high framerate, but rather algorithms for reproducing input behavior on a slightly longer term. Also, the models are strongly limited by the available input data. It may be that no suitable database entries exist for a particular scenario. On the other hand, if a database is too large, searching for the right entries may be too time-consuming.

The improvements developed in the 2010s aim to alleviate these issues. The first category of work is a continuation of the traditional techniques mentioned just now. The second category uses deep learning (DL) to create a more abstract behavioral model.

6.1. Using databases and searching

Following the ideas of Lerner et al. [LCL07], Charalambous and Chrysantou introduced 'PAG crowds' in 2014 [CC14]. This technique clusters database entries to enable faster querying. At its core is a 'perception-action graph' (PAG) where each vertex is a cluster of similar database entries, and each edge is is an observed trajectory by which an agent transitions between two clusters. During the simulation, agents query this clustered data rather than the full unordered database. Also, the method uses a separate database of entries where there is no agent interaction at all. These improvements lead to a faster simulation that is qualitatively on par with the method's predecessor.

The idea of clustering input data is also explored by Zhao et al. [ZTC13]. The main difference to the PAG method is that this work uses a trained *artificial neural network* (ANN) to determine which cluster is the best match for an input situation. Also, this work actually uses *simulated* data as input instead of real-world data. As a result, Zhao et al. obtain a model that can reportedly reproduce the behavior of a simulation algorithm (in this case ORCA [vdBGLM11]). The same ANN-based approach could apply to non-synthetic input data as well, although it will likely require a lot of input before it can make reliable decisions.

A database of agent-centric behavioral data requires a way to summarize each entry, i.e. to describe the local 'context' of an agent A_j. Boatright et al. [BKSB15] experimented with a different context description, based on the local crowd density and crowd flow in four main directions around A_j. They used a crowd-simulation algorithm to generate synthetic data, which they grouped into examples of similar contexts. They then used this grouped data to

learn (via machine learning) a behavioral policy that replicates the initial algorithm. Thus, instead of querying a database of entries at runtime, they only queried a behavioral model that was learned from all entries beforehand.

Ren et al. [RXX*21] have recently formulated crowd simulation as a data-driven optimization problem. Their method uses a database where each record contains the velocity (and several other properties) of one agent. In each simulation frame, the task per agent is to find a velocity among input data that optimizes certain criteria. These criteria are modelled via an energy function with components for e.g. collision avoidance and goal reaching. Note that the *interactions* between agents follow purely from this energy function and not from the input data itself. Thus, the method is conceptually close to a velocity-based algorithm with a cost function (as in Section 4), with the main difference that input data now determines which velocities are considered as candidates.

6.2. Using data and deep learning

For methods that use a database of crowd motion, a recurring problem is that the chosen input record will be used with little to no adaptation, thus requiring suitable records for each possible occasion. A recent trend for addressing this is to use the generalization capabilities of *deep learning (DL)*, to learn a more abstract model of agent behavior that can theoretically be applied to new situations as well. This also resolves the issue of having to search in a database at runtime, because the agent's action is now determined by the behavioral model. The work of Boatright et al. [BKSB15] was a first step in this direction, but the rise of DL techniques made this angle of approach much more popular.

This section will zoom in on the use of recurrent neural networks (RNNs), a data-driven DL technique that has recently proven to be useful for local navigation. Generally, an RNN learns to compute an expected future state relative to recent observations. In our case, an RNN estimates the next position(s) of an agent in response to its neighbors and its own past motion. Most RNN-based models come from the field of computer vision, where the purpose is tracking or human-trajectory prediction (HTP) rather than crowd simulation. However, these models are also usable for our purpose. Once trained, the RNN can immediately serve as an agent-navigation model in a crowd simulation. The navigation itself will be computationally efficient and can be used for many agents in real-time.

Research on HTP is advancing rapidly, and it seems very much detached from traditional crowd-simulation research. For example, one of the most-cited papers in the domain [AGR*16] only mentions social forces as the main method for crowd simulation, seemingly unaware of all the other developments that we are discussing in this survey. Conversely, it is possible that we (coming from the computer-graphics area) are also missing a higher-level overview of *their* domain. To the best of our ability, we will now briefly discuss a number of publications that appear to be particularly useful for crowd simulation, and we will mention their overall advantages and disadvantages. For a more complete overview of HTP research, we refer the reader to a recent survey by Rudenko et al. [RPH*20].

The Social-LSTM approach by Alahi et al. [AGR*16] uses LSTMs (Long Short-Term Memory), a type of RNN that can learn both long-term and short-term patterns in data. Using several popular real-world datasets, the authors show that their LSTM-based method is good at predicting how the input trajectories of people will continue, and that it is (understandably) more accurate than force-based methods that are not based on this input data.

A downside of LSTMs is that they always make one single prediction, which roughly corresponds to the *average* behavior that has been observed among input samples. One way to overcome this is to use a Generative Adversarial Network (GAN), a type of model that can generate a multitude of outcomes (or more specifically, a probability distribution of results). In general, a GAN consists of two parts: a *generator* that should generate new data based on input data, and a *discriminator* that should assess whether any given data is real or fake. Both of these parts are based on neural networks, and more specifically on LSTMs in our case. The generator and discriminator 'compete' during the training phase, which ideally results in a generator that is increasingly convincing.

Gupta et al. [GJF*18] and Amirian et al. [AHP19] have recently presented GAN-based methods for trajectory prediction, with minor differences between them that are too subtle to discuss here. Compared to an LSTM-only approach, these methods can generate a greater variety of trajectories using the same input. On the other hand, the training process of GANs is time-consuming and hard to control. Both articles describe to some extent how the results improve during training, and how the model is affected by parameters. Still, the training process and the optimal parameter settings may be different for each dataset.

6.3. Using reinforcement learning

In reinforcement learning (RL), a system learns to achieve an objective via trial and error. The two key components of an RL system are the *state description* that encodes an agent's current situation, and a *reward function* that rewards or penalizes certain actions (that lead to a change in state). By learning how these rewards accumulate, the system learns what the best short-term actions are to achieve long-term goals. RL is traditionally used for tasks where the goal is clearly defined (e.g. reaching a target position or winning a game) but where reference data does not necessarily exist.

Compared to LSTMs and GANs, RL produces models that optimize objective criteria, instead of models that produce similar behavior to input data. Thus, RL is not generally seen as 'data-driven', except for instance if 'similarity to input data' is part of the objective (and therefore part of the reward function). Despite this conceptual difference, we still treat RL in this section, also because we expect that future work along these lines will become more data-driven. (For example, there is an upcoming subdomain of *inverse RL* where the purpose is to infer an RL model from data.)

We will briefly discuss recent articles that apply RL to local navigation. Note that the subdomain of *multi-agent RL*, where agents collaborate to achieve a task, is beyond the scope of our survey.

In 2015, Casadiego and Pelechano [CP15] showed preliminary results of an RL method for agent navigation. Their state description encodes the relations between an agent, its goal, and its neighbors in a discretized way. Their reward function rewards getting

closer to the goal, and it penalized getting too close to obstacles. This already leads to promising results, but the authors note that finding the 'best' problem design is difficult, and that this design process should preferably be automated.

In light of this, *deep* reinforcement simplifies this modelling by using a deep neural network (DNN). Although a 'hand-made' reward function remains necessary, the state description can now be raw data (e.g. the relative positions and velocities of neighbors) instead of a tailor-made summary. After training, the DNN can compute the attractiveness of any candidate action in a given state. In 2017, Chen et al. [CLEH17] implemented such a system in the context of robot navigation. Given a state, their DNN estimates the time to the goal for any candidate velocity $\mathbf{v}'$. The network is trained using trajectories produced by ORCA [vdBSGM11] as input. After training, the method often sends agents to their goals more efficiently than ORCA itself. The resulting model is actually close to a velocity-based method (Section 4), but with the velocity cost function 'hidden' inside a DNN that has been trained via RL.

Lee et al. [LWL18] applied a similar technique specifically to crowd simulation. Haworth et al. [HBM*20] have explored a different learning mechanism, and they applied it to footstep-based navigation. Although their state descriptions and reward functions are slightly different, the shared conclusion is that deep RL with a simple reward function can outperform traditional methods.

6.4. Summary

Data-driven crowd simulation started by simple ways to automatically copy and paste the 'best' input behavior into a given context. This was followed by new ways to search the input database and to reason about its patterns. Most recently, the rise of deep learning (DL) has led to more abstract data-driven or learning-based models that do not explicitly reason about behavior anymore.

A recurring point of criticism regarding DL (in any application area) is that the trained model is a black box that no longer has any intuitive meaning in the original domain. A DNN may produce convincing results, even in unforeseen scenarios. However, it is not intuitive *why* such a model behaves the way it does, and it is not possible to make any *adjustments* to improve the behavior in individual cases. Also, it may be difficult to *combine* DL models to achieve combined effects, in contrast to e.g. combining collision avoidance with group behavior in a traditional way. To make DL-based crowd simulation more promising on the long term, researchers need to find ways to interpret the produced models.

On the other hand, DL can be a powerful way to simulate those behaviors that cannot be captured in rules alone. We also believe that DL can and will play *other* important roles in future crowd-simulation research, outside the context of learning a behavioral model. We will return to this matter in Section 12.

7. Other types of local behavior

Section 1 explained that microscopic crowd simulation divides the behavior of an agent into multiple 'levels'. Most research has focused on the local level: the tasks that each agent executes in every frame of the simulation, such as collision avoidance. This has been the primary focus of our survey so far. Still, for completeness, it is useful to reflect on how the *other* aspects of microscopic crowd simulation have evolved in the 2010s. In this section, we will look at two other types of *local* behavior that have received substantial research attention in the 2010s: group behavior and following behavior. Compared to the data-driven models of Section 6 (where all types of local behavior were merged into one), we now return to 'traditional' modelling based on rules and functions.

7.1. Group behavior

A popular type of additional behavior for crowd simulation is *group behavior*: the formation and motion of (small) groups resulting from the social relations between people in the crowd. In a crowd simulation with both global and local planning, a group typically shares a common global path, and the group's behavior is then modelled purely locally. With his pivotal work on flocking and steering behaviors, Reynolds [Rey87, Rey99] was potentially the first to describe simple ways to model how moving entities stay together. Later, more methods appeared that are designed specifically for social groups in human crowds.

Force-based modelling of groups is particularly popular because force-based simulation models are already the most common choice in applied literature. Generally, agents can receive an additional force that attracts them to the center of a group or that makes them follow a leader. There are many articles that present slight variants of this concept. In line with the rest of this survey, we will not discuss such literature, and we will instead focus on work from the 2010s that models groups in significantly new ways.

The work of Moussaïd et al. [MPG*10] and of Karamouzas and Overmars [KO12] aims to model the formations of small groups of typically 3 or 4 pedestrians. In the real world, such groups often take a V or U shape to facilitate communication. Moussaïd et al. do empirical observations to reach this conclusion, and they present a way to model such shapes such via forces. Karamouzas and Overmars let a group evaluate (in each simulation frame) the possible formations that it can attain, so that this formation can change dynamically whenever this is necessary due to other agents or obstacles. Their method first chooses an optimal formation and velocity for the whole group. Next, they compute a preferred velocity for each agent so that this formation is maintained. These preferred velocities are then used as input for collision avoidance.

In the context of slightly larger groups, Qiu and Hu [QH10] introduced a new way to define the relations inside a group of k agents. Their method uses a $k \times k$ matrix where each cell contains a scalar between 0 and 1. The number in a cell (i, j) indicates how strongly the jth agent in the group wants to follow the ith agent. Different (user-specified) matrices will lead to different kinds of group formations, such as a linear chain of agents or a cluster of agents following a leader. This can be seen as a generalization of one of Reynolds' steering behaviors [Rey99], with more control over exactly which agents influence another agent. The same concept can also be used to define relations *between* groups.

In 2017, Ren et al. [RCB*17] translated group behavior to the paradigm of *velocity obstacles*. Recall from Section 4 that, in the context of collision avoidance, a velocity obstacle (VO) is a set

of 'forbidden' velocities that will let an agent A_j collide with a particular neighbor A_k in the near future. To model how agents stay together, Ren et al. introduced the converse concept of a *velocity connection* (VC): a set of velocities that will let A_j stay sufficiently *close* to a neighbor A_k. By combining VOs and VCs, A_j can look for a velocity that avoids imminent collisions *and* satisfies grouping constraints. In practice, this is implemented via a cost function and sampling (as in Section 4.1), where the cost of a velocity $\mathbf{v}'$ now also depends on the distance of $\mathbf{v}'$ to all VCs. Thus, the result is a velocity-based algorithm that combines collision avoidance and group behavior in one function.

Kremyzas et al. [KJG16] focused on scenarios with dense crowds or obstacles that can (temporarily) break the coherence of a group. Their method roughly defines a group as coherent if all members can see each other and if they are all moving in the same direction. The method uses a notion of leaders and followers, but the leader is dynamically defined as the agent who has progressed furthest along the group's global path. Agents switch between specific behaviors based on the cohesion state of their group. For example, if the group gets disconnected, the leader can wait and the followers can plan a personal global path to the leader's position. As such, group behavior can also affect the *global* paths of agents, so grouping now acts on multiple levels of navigation. In theory, any crowd simulation with 'classical' group behavior can be extended with this functionality. In exchange, of course, the model becomes more complicated overall.

Finally, a slightly different line of work is to model how groups of agents emerge automatically, without explicitly modelling beforehand which agents belong together. Lemercier and Auberlet [LA15] empower agents with higher-level reasoning to let them choose specific interactions with their neighbors (such as avoidance or grouping) according to the situation at hand. He et al. [HPNM16] explored the same idea specifically for the emergence of groups. For any groups that are formed dynamically, their method adds collision avoidance between these groups using a velocity-obstacle approach. Both of these articles show that dynamic grouping can make the crowd's motion more efficient or more human-like according to certain metrics. However, note that they model a different kind of grouping than the other articles in this section. For social groups of agents that stay together during the simulation, it is more intuitive to model these groups explicitly.

7.2. Following behavior

Another frequently-studied aspect of human locomotion is how people *follow* each other in (for example) a corridor or a queue. Specifically, a person controls its *speed* in a particular way when it is walking behind another person. This speed control can lead to a typical higher-level 'stop and go' phenomenon of acceleration and deceleration that propagates through a crowd. Standard local-behavior algorithms do not explicitly model this speed adaptation; they simply let each agent move towards a goal while avoiding collisions, and any acceleration or deceleration that follows from this is implicit. In the 2010s, several researchers have studied following behavior experimentally, and they have attempted to translate their findings to extra rules for a crowd-simulation model.

Lemercier et al. [LJK*12] describe controlled experiments in a circular queue, where they have measured the speed adaptations of individual people. They summarize this in an equation where a person's acceleration depends on the crowd density, the difference in speed with the person in front, and a delay time. Finally, they show that this additional rule brings a crowd simulation (using RVO) closer to these real-world measurements.

Rio et al. [RRW14] extended the view from 'one-dimensional' queuing to 'two-dimensional' following behavior where the followed person moves freely through space. They describe an experiment where one person (the leader) walks around in an empty space, and another person (the follower) is instructed to follow the leader while keeping a fixed inter-personal distance. They also perform an experiment in virtual environment where the follower is a real person and the leader is artificial. For one-dimensional following (as in queuing), they show that a simple speed-matching model matches their data sufficiently well; more complicated models do not give better results. For two-dimensional following, they suggest to control the speed and angle separately, just like in several collision-avoidance algorithms. Recent retina-based methods [DMCN*17, LCMP19b] also show (as a side result) that they can achieve following behavior with a slightly adapted cost function.

In 2018, Warren et al. [War18] studied following behavior in larger crowds where leaders and followers are not pre-defined. Using a virtual-reality set-up, this work empirically explores the question of 'who follows who'. This leads to a more detailed simulation model where an agent A_j chooses specific neighbors to follow.

8. Advanced navigation models

In this section, we discuss various other models that aim to improve the navigation of agents in crowds. In a multi-level crowd simulation, the methods in this section do not (only) operate on the local level of agent behavior. Instead, they deal with other concepts such as path following, more detailed navigation around agents, or global path planning around obstacles.

The methods discussed here are more dispersed and more difficult to categorize, and a full analysis can quickly get lost in details and disrupt the focus of this survey. We believe that it suffices to give a broad overview of each method and its position in the overall concept of microscopic crowd simulation.

8.1. Detailed collision prediction

Collision-avoidance algorithms usually assume that agents will follow a linear trajectory for some amount of time, and they focus only on the avoidance of agents that are nearby. As mentioned earlier, these limitations are compensated by the fact that the algorithms are re-performed in each simulation frame. However, some scenarios may still require a more advanced type of collision prediction. Several researchers have addressed this in the 2010s. Note that some data-driven algorithms from Section 6 also revolve around detailed trajectory prediction. We will not discuss these methods again here.

Golas et al. [GNCL13] have explored the concept of *long-range collision avoidance*. Here, an agent A_j performs collision avoidance at different levels of detail for different amounts of time in the future. For the levels that are further away, agents are clustered

into larger abstract entities to avoid, and the radii of these clusters are reduced to represent the uncertainty of the crowd's future state. This technique can improve a simulation where agents would otherwise respond too late to large groups of agents that are far away. The authors perform experiments with RVO as the main collision-avoidance algorithm, but the level-of-detail concept can be applied to other algorithms as well.

The 'WarpDriver' method by Wolinski et al. [WLP16] improves upon the collision-avoidance assumption that agents follow linear trajectories. Based on the layout of the environment and on the movement of agents in the recent past, the method can make more advanced predictions of where all agents will end up at certain moments in the future. These predictions are done via so-called 'warp operators' that map the motion of agents to 'collision probability fields', which indicate how the risk of collision with an agent spreads throughout the environment over time. The authors show that this technique can handle many scenarios for which traditional collision avoidance is too simplistic, including crowd simulation in curved corridors and the avoidance of zig-zagging agents. So far, WarpDriver appears to be one of the last 'non-deep-learning-based' methods to reach this level of abstraction.

8.2. Detailed navigation around agents

A related but slightly different idea is to change the way in which an agent moves around its neighbors. In earlier work by Kapadia et al. [KSHF09], each agent computes a short-term motion plan where the speed and direction can change over time. Likewise, in the same group's work on footstep-based planning [SKRF11], agents plan a short-term sequence of footsteps, which may contain more complex interactions than in a simple collision-avoidance algorithm.

A method by Godoy et al. [GKGG14] lets an agent try out multiple directions of motion and estimate a future scenario for each direction. The agent then chooses the best direction based on 'hindsight optimization' (HOP), using a reward function that prefers both progress towards the goal and low energy consumption. Technically, this algorithm updates the *preferred velocity* $\mathbf{v}_{\text{pref}}$ of an agent, which can then be fed to any collision-avoidance algorithm. The inclusion of HOP can improve the results in (for example) symmetric scenarios where collision avoidance only would lead to indecisive behavior or deadlocks.

Bruneau and Pettré [BP17] have presented another algorithm that operates on the preferred-velocity level. They consider so-called *mid-term planning* where an agent A_j explicitly plans a *sequence* of upcoming avoidance actions around other agents. An avoidance action can be to move around an agent along the left or right, or to speed up or slow down. By planning multiple of these actions in a row, A_j can (for example) navigate through a dense crowd with a non-linear 'corridor' of free space inside it. Such a scenario cannot be resolved by a collision-avoidance algorithm that only extends the motion of A_j linearly.

Recent work by Mavrogiannis et al. [MTK18, MK19] also models explicitly how an agent avoids another agent along either the left or right side. Furthermore, agents can *communicate* this topological decision to each other, so that they can quickly agree on a combined strategy. For scenarios with more than two agents, the authors study all possible combinations of topological decisions based on braid theory. This rapidly increases in complexity, which makes it less suitable for real-time simulations of large crowds. However, just like retina-based collision avoidance, it is very useful in the context of robot navigation and human-robot interaction, which is the context on which Mavrogiannis et al. focus.

The work on 'torso crowds' by Stüvel et al. [SMTTvdS16] focuses on navigating through dense crowds of (mostly) stationary agents, such as in an elevator or in a crowded bus. This method models agents as 2D capsules and uses a Voronoi diagram (VD) to describe the free space between agents. A moving agent repeatedly looks for a path through this VD that is either short or comfortable. Meanwhile, stationary agents can move and rotate to make space for the moving agent, or to improve their own comfort in the crowd. Compared to classical crowd simulations, this is a more detailed representation of the agents and of the space between them.

Most of these methods blur the traditional line between local and global behavior, by modelling explicitly how to navigate around agents, thus almost treating agents at the same level as obstacles. This may be more reliable than repeated local decision-making only. Of course, it comes at a computational cost, which limits the number of agents that can be simulated in real-time.

8.3. Path following

Section 2.2 explained that the *preferred velocity* $\mathbf{v}_{\text{pref}}$ of an agent is not necessarily a velocity that points straight to the goal. It can also be a velocity that lets the agent follow a path that has been computed via global path planning. This is crucial in complex environments with obstacles (such as the interior of a building) where navigation is not a purely local concept.

The idea of *path following* was already described informally by Reynolds in 1999 [Rey99]. In 2009, Karamouzas et al. [KGO09] made this more formal with their *Indicative Route Method (*IRM*)*, where an agent A_j always computes an *attraction point* $\mathbf{p}_{\text{att}}$ along the path π that can be reached by a straight line. The preferred velocity $\mathbf{v}_{\text{pref}}$ is then the vector that sends A_j to $\mathbf{p}_{\text{att}}$ at a certain preferred speed. However, this method relied on a particular data structure (the medial axis) for global path planning, so it was not yet easily usable in any crowd simulation.

In 2013, Jaklin et al. [JCG13] presented MIRAN, an alternative path-following algorithm that lets an agent evaluate multiple 'candidate' attraction points along π and choose the best according to certain criteria. This paper also introduces the concept of a *reference point* $\mathbf{p}_{\text{ref}}$ that indicates how far an agent has progressed along its path. Finally, the criteria for choosing between candidates can also be based on *weighted regions* in the environment, i.e. different terrain types that are less or more attractive to walk through, such as sidewalks, grass, or mud. MIRAN can be used in any crowd-simulation system regardless of how global planning is handled. However, compared to its 'predecessor' IRM, MIRAN adds the computational cost of evaluating several candidate points.

In recent work that primarily deals with data-driven global paths, Amirian et al. [AvTHP19] give each agent a path that contains *time* or *speed* information. Such a path is not just a curve, but it also

specifies how fast an agent should move along it, or at what time the agent should reach certain points. This asks for a path-following algorithm that lets agents speed up, slow down, or stand still when its input path demands this, while allowing for collision avoidance at the same time. The authors propose a preliminary algorithm where an agent is always attracted to a point that lies a number of seconds in the future. This works sufficiently well at the low crowd densities of their input data. However, when multiple agents want to occupy the same space at the same time, the desired behavior of the crowd is not well-defined. This topic still requires further research.

8.4. Adaptation to density or flow

Some algorithms from the 2010s update the preferred velocity $\mathbf{v}_{\text{pref}}$ for an agent A_j based on the surrounding density or flow. This updated $\mathbf{v}_{\text{pref}}$ will then be used as input for collision avoidance.

One method in this category is *DenseSense* by Best et al. [BNCM14]. It is based on the *fundamental diagram*: an empirically established relation between the crowd density and the typical walking speed of people. The algorithm lets an agent A_j evaluate several alternative directions around $\mathbf{v}_{\text{pref}}$ via sampling. For each candidate direction $\mathbf{d}'$, A_j computes the crowd density ρ' for a point in the future, and it then computes a corresponding velocity $\mathbf{v}'_{\text{pref}}$ whose speed matches ρ' in the fundamental diagram. Finally, A_j chooses the candidate velocity with the smallest expected time to reach the goal. As a result, an agent can decide to move around dense areas, or (if not) it will at least adapt its preferred speed to the upcoming density. The authors show that this yields a speed-density relation that is closer to real-world fundamental diagrams, regardless of the collision-avoidance algorithm that is used. On the other hand, in multi-directional pedestrian flows, it may not be sufficient to base decisions purely on the crowd *density*.

Heliövaara et al. [HKHE12] had previously proposed a model specifically for multi-directional crowd flows. Its description is built around the social-force model, but its main contribution is actually a separate step that bears similarities to *DenseSense*. In each frame, an agent A_j evaluates multiple possible directions centered around the preferred velocity, and it chooses the option that (roughly speaking) offers the least resistance in terms of opposing crowd flows. The authors show to what extent this model makes the crowd's behavior more realistic or human-like. A combination of this technique with *DenseSense* would be interesting to investigate, as it would take both density and flow into account.

Another method that updates the preferred velocity based on crowd flow is *Stream* by van Goethem et al. [vGJCG15]. It intends to model how a person increasingly 'goes with the flow' as the local crowd density increases. Each agent A_j adapts its velocity in two ways: to *match* the velocity of neighbors that are nearby, and to move *towards* neighbors that are farther away. All neighbors are combined using an appropriate weighting scheme, while ignoring any neighbors that go in the complete opposite direction. The result is a 'perceived stream velocity' $\mathbf{v}_{\text{str}}$ for A_j. Finally, A_j updates its preferred velocity as an interpolation between $\mathbf{v}_{\text{str}}$ and its original $\mathbf{v}_{\text{pref}}$. This interpolation is based on an *incentive* parameter that models an agent's willingness to go with the flow. The authors show that this model can (for example) help prevent deadlocks at bidirectional crowd flows in a narrow corridor. The method is computationally cheap because it only combines the velocities of neighboring agents, without requiring any sampling. However, *Stream* is not based on real-world findings such as the fundamental diagram.

Recall that several mid-term planning algorithms of Section 8.2 *also* update the preferred velocity, but for a different purpose and in a different way. In other words, both concepts occupy the same position in the usual multi-level framework for agent navigation. If a simulation engineer would like to benefit from the advantages of both concepts, it is not entirely clear how they should be applied in sequence or combined into one process.

The *BioCrowds* model by De Lima Bicho et al. [dLBRM*12] is difficult to categorize, but we mention it here because it yields density-dependent behavior as well. In this model, the obstacle-free part of the environment is densely sampled with 'marker' points. At any moment, each agent claims the markers to which it is closest, resulting in a sampled version of the Voronoi diagram of all agent positions. Each agent then moves towards a weighted average of its own markers, where the weight function contains a goal-reaching term. The authors use this as the entire simulation model, arguing that collisions do not occur because each agent stays inside its own Voronoi cell. However, to obtain more advanced agent behavior, we argue that the model should be combined with other techniques from this survey. We believe that *BioCrowds* could act as a density-dependent filter in a larger system, similarly to *DenseSense* or *Stream*. After all, the use of Voronoi cells causes agents to be nudged in a direction that offers more free space.

8.5. Combination with global path planning

Traditionally, global navigation (path planning) and local navigation (e.g. collision avoidance) are treated as two separate steps of a crowd simulation. Global navigation is particularly important in large environments with complicated geometry, such as buildings or cities. The seminal work of Shao and Terzopoulos [ST07] was one of the first successful examples of such a large-scale simulation where agents repeatedly make both global and local decisions.

This separation of tasks works well in many scenarios, but there are cases where it may not suffice. For example, many agents may end up using the same parts of the environment at the same time, while alternative paths remain unused. Also, collision avoidance may send an agent into a part of the environment that does not match its current global path, which can lead to erratic and indecisive behavior. There are several ways to address such issues.

We acknowledge that path planning is a large research area that could easily warrant a state-of-the-art report of its own. Also, most work on path planning is conceptually detached from crowd simulation. In this section, we will only look at developments of the 2010s that are specifically meant for crowds and that solve problems specific to this domain. This excludes general work on path planning amidst obstacles, as well as work on 'multi-agent path planning' that typically assumes only a small number of agents.

8.5.1. Density-based path planning

In 2010, Höcker et al. [HBK*10] have combined graph-based path planning with crowd-density information. For every edge in an ar-

bitrary graph, they measure the crowd density in a surrounding rectangle and store this in the corresponding edge. Agents can use this data to not plan a *shortest* path in the graph, but a (predicted) *fastest* path, based on the fundamental diagram that translates crowd densities to typical walking speeds. This results in agents automatically taking detours if this leads to an expected time gain.

In 2012, van Toll et al. [vTCG12] applied the same concept to a *navigation mesh*, which is a path-planning data structure based on non-overlapping polygonal regions. For each region of the navigation mesh, they store to what extent it is occupied by agents. This yields a coarse representation of the crowd density throughout the environment. They then apply the same 'fastest-path algorithm' by Höcker et al. [HBK*10], with an additional parameter for making agents less or more sensitive to delay. The authors show that this can improve the crowd flow in environments with multiple alternative paths of roughly equal length. However, the method's density representation is very coarse, and it depends on the shapes and sizes of navigation-mesh regions. For example, agents cannot detect small congestions inside large regions of the navigation mesh. It would therefore be interesting to combine this idea with a method like *DenseSense* [BNCM14], which also performs density-based 'path adaptation' but on a more local scale.

Density-based path planning has a few general limitations: it is *reactive* (agents only respond to the environment's *current* density distribution), and there is no true collaboration between agents. In exchange, though, it is conceptually simple and does not require much computational overhead.

8.5.2. Space-time planning and coordination

Another way to improve coordination in the crowd is to let agents plan their paths in both space and time. Roughly, this means that the usual planning domain (a graph or a navigation mesh) is extended with a time dimension. Different agents can ensure that they will not occupy the same critical area at the same moment. Space-time planning is a broad topic that goes beyond crowd simulation. We will now touch upon the work that applies specifically to crowds.

In 2011, Singh et al. [SKH*11] integrated space-time planning into a multi-level agent navigation framework. They showed examples of narrow corridors where agents explicitly wait for other agents to pass. Such behavior cannot easily be modelled in a traditional crowd simulation. Of course, space-time planning does come at a computational cost. This specific work uses a grid-based environment representation that is less efficient than a navigation mesh.

The work of Lopez et al. [LLL12] and Kapadia et al. [KBG*13] is intended for fully animated virtual characters that can jump and climb, and for environments that can change over time (with e.g. moving platforms). These papers show examples with multiple agents, but the methods are not necessarily intended for typical crowd-simulation scenarios, where the real-time simulation of large crowds is more important than the capabilities of each agent.

One noteworthy method by Karamouzas et al. [KGvdS12] uses the space-time concept specifically to guide entire *crowds* through an environment with obstacles. They use linear-programming techniques to plan globally coordinated trajectories for groups of agents, so as to minimize the average travel time per agent. This coordinated plan can also include *waiting* behavior where one group explicitly waits for another group to pass. The authors show that this can improve the overall crowd flow in several bottleneck-like scenarios. Such a coordinated method partly makes the simulation *macroscopic* as agents do not purely think for themselves anymore. A potential risk is that the crowd may become 'overly coordinated' compared to real-world behavior where any coordination emerges dynamically. For the same reason, though, the technique could be very useful for crowd-management purposes, to steer people in the right direction depending on a globally coordinated plan.

8.5.3. Communication between navigation levels

In 2019, van Toll and Pettré [vTP19] studied the problem where local navigation sends an agent in a direction that does not match its global path. Their method defines a *topological strategy* as a set of decisions to move around agents/obstacles along the left or right. (As discussed in Section 8.2, Mavrogiannis et al. [MK19] explored similar ideas for collision avoidance among a small number of agents.) Such a strategy can be computed for the agent's global path, for its preferred velocity that results from path following, and for the velocity that results from collision avoidance. An agent can then periodically check if the strategies of these different levels are in conflict, e.g. if two algorithms are trying to send an agent around an obstacle in opposite ways. If such a conflict occurs, the agent can make explicit decisions, such as choosing a new velocity that *does* follow the global path, or *re-planning* a global path with the detour suggested by collision avoidance. On a critical note, this concept relies strongly on the individual navigation algorithms being sufficiently 'intelligent' to even yield alternative strategies.

This work was extended in 2020 [vTP20] to a more abstract framework where all levels of navigation can communicate. The framework has only been implemented to a certain extent; for example, a mid-term planning algorithm [BP17] could still improve the results greatly. Also, on the long term, it would be conceptually stronger to merge all levels of navigation into one hybrid process, instead of keeping the levels detached and solving conflicts.

8.6. Summary

In this section, we have discussed many agent navigation algorithms that operate on a different level than local behavior, such as path following or global path planning. As long as these algorithms have a clear place in the overall crowd-simulation system, they can be arbitrarily combined.

Each additional algorithm is often designed around specific scenarios where a 'minimal' simulation (with only path planning and collision avoidance) does not work. Within the context presented in each individual paper, it is clear that the corresponding new algorithm improves the results. However, our research area still lacks a more *general understanding* of which algorithms work best in which scenario, let alone a way to reason about this automatically. New insights in this regard could help create an 'ultimate' crowd-simulation system that chooses the right algorithm at the right time. We will explain in Section 12 that this is one of the main directions for future work that we envision.

9. Frameworks and implementations

The increase in research on microscopic crowd simulation worldwide has led to the development of several software frameworks throughout the 2010s. These have been described in technical reports or other publications. We will briefly outline them here.

The *Nomad* model, developed by transportation researchers at TU Delft since the early 2000s, uses force-based models to simulate the behavior of agents. Many of its content is based on real-world measurements of pedestrian traffic in urban environments, such as public-transport facilities. It focuses mostly on an elaborate model per agent and on matching with real-world data, and less on real-time performance for large crowds. A report from 2014 [CHD14] described the latest developments of *Nomad* at the time.

Most other frameworks are from the computer-science community and focus on efficiently simulating large crowds, usually in exchange for a simpler agent model. The work by Singh et al. [SKH*11] discussed how to combine different levels of behavior, such as path planning, collision avoidance, and contact forces. Later, several other research groups have presented frameworks based on a similar subdivision into modular levels, but with more implementation details, and often with freely available source code. Examples include the *Explicit Corridor Map* framework from Utrecht University [vTJG15], *Menge* from the University of North Carolina at Chapel Hill [CBM16], *MomenTUM* from the Technische Üniversität München [KBB16], and *Vadere* from Munich University of Applied Sciences [KZGK19]. Each framework has its own focus area depending on the founding group's expertise, but the overall concept is always similar.

The ADAPT framework from the University of Pennsylvania [SMKB13] has a broader scope that also includes the detailed 3D animation of individual people. In contrast to 'pure' crowd-simulation frameworks, ADAPT is meant as a prototyping and experimentation tool for any research that involves animated virtual humans. For navigation and collision avoidance, it uses the built-in functionalities of the Unity3D game engine.

The recent UMANS framework of Inria Rennes [vTGL*20] focuses purely on the local aspect of navigation. It translates many of the algorithms from Sections 3–5 to the common concept of using a cost function in velocity space, while keeping other simulation details uniform and separated. As such, UMANS can serve as a basis for an objective comparative study of collision-avoidance algorithms. However, it (deliberately) does not provide a complete simulation solution for environments with complex geometry.

The developments in academia have also led to various *commercial* software solutions for microscopic crowd simulation; we will not list these here individually. These solutions often also contain useful tools for intuitive scenario construction and output analysis, and optimizations for simulating large crowds in real-time.

Overall, many frameworks for microscopic crowd simulation exist, often based on similar principles but with different details and focus areas. The multi-level approach to crowd simulation allows new algorithms to be easily integrated, as long as they clearly operate on a particular level (such as local behavior). At this point, we argue that there is no need for even more of these frameworks. Instead, the research area would greatly benefit from a better behavioral analysis of the algorithms *inside* these frameworks, which requires new concepts for reasoning about the quality of an algorithm. The research in this area is still in an early stage, and it is made more difficult by the fact that crowd simulations are complex systems, where small implementation details can have large effects.

10. Evaluation of simulation results

The *evaluation* of crowd simulations can concern different aspects of an algorithm: its real-time performance, its ability to compute energy-efficient motion of agents, or (most importantly *and* most abstractly) its level of realism. To demonstrate superiority over previous techniques, each paper of a new simulation algorithm incorporates some form of evaluation of the simulation results or algorithmic properties. Nevertheless, the question of *how* to properly evaluate simulation results is challenging enough on its own. It has received a significant amount of research attention, especially in the last decade. Although this topic lies at the far edge of our survey's scope, it is useful to mention the main developments.

Earlier work focused on developing *metrics* for describing the quality of a simulation. The notion of quality may differ per application. For example, entertainment applications usually aim for visual realism, and they will want to avoid sudden accelerations, oscillations, or collisions between agents. In 2009, Singh et al. [SKFR09] presented a benchmark framework that contained mostly metrics of the first category. Given the output of a simulation in a certain scenario, this framework could measure e.g. the time for agents to reach a destination, the total energy spent by agents, and the number of collisions. These results could then be combined into a benchmark score with user-specified weights.

Conversely, safety applications will be more interested in the predictive capabilities of a system: how does a simulation compare to reality in terms of, for example, flow capacity or evacuation time? Most of these concepts are highly scenario-specific. An important general concept is the *fundamental diagram*, which was first described for pedestrian traffic in 1993 [Wei93]. As outlined earlier, the fundamental diagram is the average relation between crowd density and walking speed, and it has been measured in many real-world scenarios. For safety-critical applications, it is important that a simulation algorithm can faithfully reproduce this speed-density relation. Many of the papers that we have discussed therefore include a certain comparison to the fundamental diagram. However, this diagram is only known for a number of 'classical' scenarios such as corridors and T-junctions. If a simulation algorithm can reproduce the fundamental diagram in simple scenarios, this does not guarantee that the algorithm remains 'realistic' in other cases.

In the 2010s, the research on evaluation concentrated on three aspects: calibrating simulation parameters, comparing simulations to real data, and gaining deeper knowledge of the possible scenarios that can occur. We will treat these topics in separate subsections.

10.1. Parameter calibration

As the results of a simulation depend on its input parameters, an important part of obtaining high-quality output lies in choosing the

right parameter values. This parameter tuning is often done manually. However, two articles from 2014 revolve fully around *automatically calibrating* simulation parameters to optimize certain criteria [WGO*14, BKHF14]. Both of these papers pose this as an optimization problem. Given a metric $\mathcal{M}$ for simulation quality and a set of parameters $\mathcal{S}$, the problem is to find the values for $\mathcal{S}$ for which the simulation output maximizes $\mathcal{M}$. The articles differ mostly in the types of metrics on which they focus.

Berseth et al. [BKHF14] have built such an optimization framework around the SteerBench benchmarking suite [SKFR09]. This paper presents different ways to calibrate parameters in optimization of the SteerBench metrics. The authors also suggest other types of metrics, such as the similarity with 'ground-truth' data containing real-world trajectories. This similarity is a complicated concept on its own, and we will discuss it more in the next subsection.

The work of Wolinski et al. [WGO*14] focuses more on the similarity to datasets. They include both *microscopic* metrics (based on the properties of individual trajectories and the differences between them) and *macroscopic* metrics (based on overall behavior that results from the simulation, such as the fundamental diagram). They also proposed and compared several search algorithms for finding optimal simulation parameters, and they considered the option to optimize multiple metrics at the same time.

Overall, we can conclude that many crowd-simulation algorithms can be calibrated to optimize certain quality metrics. The most interesting research questions lie in which metrics to use, which scenarios to simulate, and (if 'similarity to real data' is chosen as a metric) which reference data to compare to.

10.2. Comparing a simulation with data

The question of comparing crowd-simulation results with data (usually the trajectories of a real crowd) is not a simple one. As said, analysis based on the fundamental diagram is useful as a first step, but this is limited to simple scenarios. For more complex cases, comparing a simulation to real data usually implies a comparison between the actual *trajectories* of agents and people. However, human behaviour is subject to variation, and this variability is combined with the chaotic nature of collective behaviour. Thus, on a small scale, the trajectories of humans in a crowd in similar (or even identical) situations will not show the same evolution. If two real trajectories would already diverge, what can one say about a simulation that diverges from real data? This motivated the development of specific comparison techniques that we report below.

Early work focused on very simple controllable scenarios. Aiming to validate a new velocity-based steering algorithm, Pettré et al. [POO*09] compared their results (and those of force-based algorithms) with real recorded data of two humans crossing. The dataset from this work was later re-used to evaluate the PLEdestrians algorithm [GCC*10]. However, such an evaluation is confined to the specific case under consideration, and it does not extend to general scenarios with more people and interactions.

Lerner et al. [LCSCO09] used the principles of their data-driven simulation method [LCL07] to compare a simulation to real data. Given the local state of an agent and of its neighbors, they search for the most similar entry in an input database. The evaluation then depends on the similarity between this entry and how the simulation evolves. However, just like in data-driven crowd simulation, the problem remains that a dataset will never cover all possible states and variety in human behaviour, so it is difficult to draw strong conclusions about the level of realism.

Charalambous et al. [CKGC14] followed up on this work by using an outlier-detection mechanism. Their method can identify an agent trajectory (or part of a trajectory) that is not similar to a given dataset. The outlier detection can again be based on several similarity metrics. Note that this method considers each trajectory independently, so it will not evaluate any features related to interaction.

With the aim of widening the scope of these evaluations (so as to consider interactions as well), Guy et al. [GVDBL*12] established a more general comparison metric. They explore the statistical difference between two sets of trajectories (be it real or simulated) based on entropy. To compare a simulation to reality, they set up a simulation to match an initial state provided by data (at time t), run the simulation for a short period of time (say Δt), and compute the residual error between the simulation result and the input data for time $t + \Delta t$. This comparison is performed for a large number of sampled t values, and the distribution of residues is analyzed through the eye of entropy metrics. This metric would later be re-used by the calibration work that we mentioned earlier [WGO*14, BKHF14].

While all methods mentioned so far consider only the local scale of trajectories, Wang et al. explored the features of *global* paths in given environments. In a first paper [WOO17], they proposed clustering paths to learn and discover patterns that exist among them, as well as to provide an intuitive visualization of crowd behaviors. Later, the work was extended [HXZW20] to clustering paths based jointly on shape, speed, and time features. For scenarios with many obstacles, and therefore for crowd simulations with global and local navigation, evaluation at both levels is important.

10.3. Scenarios, datasets, and coverage

We have seen that a simulation algorithm can be evaluated based on easily-measurable criteria or on the similarity to (real) reference data. In both cases, it is important to think about the scenarios that are tested, the reference data that is used, and how well these aspects cover all relevant cases.

In light of this, Kapadia et al. [KWS*11] have considered the following question: given the parameters that define the state of an agent and its neighbors, does a dataset take into account all possible combinations of these parameters? In the context of computing performance-based metrics, this work shows how to automatically generate scenarios that cover this parameter space adequately.

Karamouzas et al. [KSHG18] have worked on translating a set of input scenarios to a low-dimensional 'crowd space'. This space describes the range of possible simulation scenarios in a continuous way, based on a finite amount of input data. While this contribution is interesting on its own, the work also adds a component that estimates which crowd-simulation algorithm is most suitable for a particular scenario. The model also involves elements of deep

learning; as such, it is a good example of a DL-powered method that does not lose its domain intuition.

In terms of available data, only few real-world datasets of crowd motion exist. These have been used throughout literature for various applications such as crowd simulation, tracking, and trajectory prediction. More datasets have appeared in recent years, thanks to the effort of some research teams to acquire new experimental data, as well as to the progress in computer vision and multi-object tracking. A recent survey by Amirian et al. [AZC*20] discusses the real-world datasets that are currently available. It also analyzes the statistical properties of these datasets, to estimate how suitable they are for training a human-trajectory-prediction (HTP) system.

One idea to counter the lack of real-world data is to use generate *synthetic* data using simulations. This way, it becomes easier to create datasets that cover many possible situations, including situations that would be dangerous to orchestrate in real life. For example, the model by Boatright et al. [BKSB15] (discussed in Section 6) is trained on such synthetic data. Recently, Qiao et al. [QZK*19] have used similar data to investigate how well various data-driven models extend to scenarios on which they were not trained. However, for the more abstract purpose of obtaining 'realistic' simulation models, we argue that synthetic input data is not yet meaningful, as evaluating the realism of a model is not yet a fully solved problem itself.

10.4. Summary

The question of evaluating a crowd simulation received significant attention in the 2010s. Starting from evaluation methods based on objective criteria, we observe strong developments in terms of automating and standardizing the evaluation process, comparing to real data, and reasoning about the coverage of scenarios. However, there are still open questions, especially in the area of comparing a simulation to reality.

Given the link between evaluation and simulation parameters, it is important to mention that we have decided not to discuss any crowd-simulation work based on psychology or personality settings [DGAB15, GKLM11, KGML12]. Such work aims to capture the emotional or personality traits of real humans, or to capture the way in which people respond to each other. While this makes the comparison to real behavior seemingly easier, the underling crowd-simulation algorithms are still 'traditional', and the question of how to evaluate the results remains difficult.

Finally, as in the rest of this survey, we have limited our discussion to simulations that result in 2D trajectories. In entertainment applications, those trajectories will often be combined with full-body character animations. These animation aspects can be evaluated via e.g. perceptual user studies [HOKP16, MLC*08, MMON10]. This topic lies beyond the scope of this survey.

11. Conclusions

Simulating the motion of human crowds in real-time is useful for many applications. Most research in this area focuses on *microscopic* crowd simulation (i.e. simulating each person as an individual agent), and more specifically on the *local behavior* and *collision avoidance* per agent. This state-of-the-art report has reviewed the advancements in microscopic crowd simulation since 2010.

In the area of collision avoidance, the focus has moved from simple force-based models (founded in the 1990s) to more complicated velocity-based approaches (around 2010) and detailed retina-based methods (in the later 2010s). Another significant development is that of data-driven methods, which directly try to *replicate* input data instead of trying to *describe* human behavior. Force-based methods remain highly popular in applied literature that simulates crowds for specific case studies. In computer-science and robotics research, though, most effort is now spent on alternative models that are potentially more intelligent or more human-like. This also includes more and more algorithms that operate on other levels of navigation, such as path following, mid-term planning, or global planning based on crowd-related information.

In terms of implementation, many frameworks exist that combine local behavior with global path planning and potentially more types of navigation. Such a modular 'multi-level' simulation loop has been described several times in the 2010s, and it appears to now be a standard that works well in many scenarios.

Evaluating the quality or realism of a crowd simulation remains difficult. However, the past decade has seen interesting developments related to parameter calibration, data-driven comparisons to real crowds, and scenario coverage. As there are still many unanswered questions, this research area will continue to grow.

In each publication that presents a new crowd-simulation component, or a new way to handle a component that already existed, the authors show an improvement of agent behavior in particular scenarios. Such contributions are valuable and they will most likely continue to appear in the future. However, on the long term, the research area would benefit most from a general understanding of which algorithms are useful in which cases. Current techniques for evaluation are not yet sufficient for obtaining such general insights.

12. Outlook on future work

We conclude this state-of-the-art report with an overview of the most important avenues for future work, and with our expectations of how these topics will evolve in the upcoming decade.

12.1. Data-driven crowd simulation

Section 6 showed that data-driven models for agent behavior have received great attention in recent years due to the rise of deep learning (DL). We expect many more developments in the years to come, but it is somewhat difficult to predict what exactly these developments will be. One possible direction would be to think of new ways to encode crowd data for a neural-network model, which would also have interesting consequences for the *analysis* of human crowds. Besides this, the area of deep learning itself may see new fundamental contributions that we cannot yet imagine, similarly to how GANs have opened up new possibilities just a few years ago.

A general risk of DL-based methods is that they abstract away any domain knowledge, leading to a model that (while convincing)

does not lead to any new insights. This problem may be temporary, as the question of how to interpret such models is a growing research topic on its own. Also, there are already examples of methods that do not suffer from this as much, by only using DL for some aspects of the solution and applying domain knowledge elsewhere [KSHG18]. To improve the understanding of DL results in our domain, more collaboration between the communities of crowd simulation and machine learning is required.

Regardless of how this will evolve, though, there are several aspects of crowd simulation where this a lack of interpretation is not necessarily a problem, i.e. where it is already interesting to replicate real-world data without necessarily understanding it. This includes the seemingly 'random' local behavior of humans that are not specifically moving to a goal, or the way in which humans navigate through a partially unknown environment. These aspects of behavior are difficult to capture in rules alone, so they may be beyond the limits of what traditional simulation techniques can handle. The abstraction capabilities of DL are very useful in this regard, and we expect strong contributions in this area in the near future.

12.2. Analysis and evaluation

We also expect strong developments in the area of simulation analysis and evaluation, again related to the rise of deep learning. Many of the recent evaluation techniques from Section 10 are already data-driven, and DL is becoming increasingly popular for strongly related topics such as human-trajectory prediction [RPH*20] and crowd-video analysis [LCW*15,GF17]. One reason why DL makes sense for the analysis of (real or simulated) crowds is that many of the involved questions are inherently 'vague' and difficult to quantify, and DL abstracts away from this issue. Another reason is (again) that traditional techniques are limited by the available data, and DL could lead to a more generalized representation of a dataset. On a different scale, it could even lead to an abstract representation of *all possible data*, without the need for manually-chosen categories of scenarios. In short, we believe that DL has the potential to revolutionize the area of crowd evaluation. However, as with simulation, we hope that this will not happen without leading to a better *understanding* of the problem. If successful models are obtained, interpreting them will become the next big research question.

Another interesting domain for evaluation is *virtual reality* (VR), which has seen significant growth in recent years. VR is a powerful tool for immersing a user into a virtual environmen that can be filled with any content of choice. This allows for user evaluations of a simulated crowd from a first-person perspective. While the idea of crowds in VR has been coined in the 2000s already [PSAB07], it has recently gained popularity thanks to technical improvements in the VR domain. So far, VR has been used in particular to analyze human behavior in crowds [BOP15, MKT*16, MBV*18, BHO*20, BGB*20], but we expect that it will soon play a role in model evaluation as well. Generally, we believe that VR is a high-potential angle of approach that will continue being explored.

12.3. Choosing and combining navigation algorithms

Sections 7 and 8 discussed many algorithms for simulating other kinds of agent behavior than collision avoidance only. Most of these methods are fully detached from collision avoidance, which makes them easy to plug into an existing system. These models for special behavior may be overly complicated for many scenarios, and a traditional crowd simulation will often work sufficiently well. It would be interesting to have a system that automatically uses the right algorithm at the right time, or that enables or disables the appropriate components per agent, based on analysis of the scenario at hand. Recent efforts by Boatright et al. [BKSB15] and Karamouzas et al. [KSHG18] (which use machine learning to estimate which algorithm to use when) are interesting steps in this direction, but the concept can be taken further, e.g. to deal with *combinations* of algorithms for different levels.

Although (deep) machine learning can help determine when to use which algorithm, we expect that analyzing the algorithms themselves will remain relevant as well. Even if we look at collision avoidance only, choosing the right algorithm for a particular purpose is a complicated and poorly-defined task. Many (categories of) algorithms exist, with clear conceptual and computational differences, but it is difficult to say which algorithm works best in a particular scenario. This is partly due to each algorithm having its own implementation with different simulation details. Generalized implementations such as UMANS [vTGL*20] can help gain insights into the true conceptual differences between algorithms.

On a different note, the modularity of navigation algorithms could also be seen as a conceptual *disadvantage*, as an agent's behavior now follows from a sequence (and/or weighted average) of processes with possibly interfering criteria. It may be better to somehow *combine* all these navigation tasks into one hybrid process. An agent would then ideally choose 'the best velocity' based on all criteria combined, such as collision avoidance, adherence of a longer-term global path, adaptation to crowd density, coordination with other agents, and so on.

Combining these observations into one statement, the ideal crowd-simulation algorithm would unify all traditional 'levels' of agent navigation into one holistic process, and automatically apply the right behavior to the right scenarios. This automatic detection also depends on developments in the area of analysis and evaluation, which we mentioned earlier as another main future-work direction. Therefore, such an 'ultimate' simulation method will most likely not be created in the near future, but we expect that the research field will take steps towards it in the upcoming years.

References

[AGR*16] ALAHI A., GOEL K., RAMANATHAN V., ROBICQUET A., FEI-FEI L., SAVARESE S.: Social LSTM: Human trajectory prediction in crowded spaces. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition* (2016), pp. 961–971. 12

[AHP19] AMIRIAN J., HAYET J.-B., PETTRÉ J.: Social ways: Learning multi-modal distributions of pedestrian trajectories with GANs. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition Workshops* (2019). 12

[AvTHP19] AMIRIAN J., VAN TOLL W., HAYET J.-B., PETTRÉ J.: Data-driven crowd simulation with generative adversarial networks. In *Proc. 32nd Int. Conf. Computer Animation and Social Agents* (2019), pp. 7–10. 15

[AZC*20] AMIRIAN J., ZHANG B., CASTRO F. V., BALDELOMAR

J. J., HAYET J.-B., PETTRE J.: Opentraj: Assessing prediction complexity in human trajectories datasets. In *Asian Conf. Computer Vision* (2020), Springer. 20

[BGB*20] BERTON F., GRZESKOWIAK F., BONNEAU A., JOVANE A., AGGRAVI M., HOYET L., OLIVIER A.-H., PACCHIEROTTI C., PETTRÉ J.: Crowd navigation in VR: exploring haptic rendering of collisions. *IEEE Trans. Vis. Comput. Graphics* (2020). 21

[BHO*20] BERTON F., HOYET L., OLIVIER A.-H., BRUNEAU J., LE MEUR O., PETTRÉ J.: Eye-gaze activity in crowds: impact of virtual reality and density. In *Proc. 27th IEEE Conf. Virtual Reality and 3D User Interfaces* (2020), pp. 322–331. 21

[BKHF14] BERSETH G., KAPADIA M., HAWORTH B., FALOUTSOS P.: SteerFit: Automated parameter fitting for steering algorithms. In *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation* (2014). 19

[BKSB15] BOATRIGHT C. D., KAPADIA M., SHAPIRA J. M., BADLER N. I.: Generating a multiplicity of policies for agent steering in crowd simulation. *Computer Animation and Virtual Worlds 26*, 5 (2015), 483–494. 11, 12, 20, 21

[BNCM14] BEST A., NARANG S., CURTIS S., MANOCHA D.: DenseSense: Interactive crowd simulation using density-dependent filters. In *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation* (2014), Eurographics Association, pp. 97–102. 16, 17

[BOP15] BRUNEAU J., OLIVIER A., PETTRÉ J.: Going through, going around: A study on individual avoidance of groups. *IEEE Trans. Vis. Comput. Graphics 21*, 4 (2015), 520–528. 21

[BP17] BRUNEAU J., PETTRÉ J.: EACS: Effective Avoidance Combination Strategy. *Comput. Graph. Forum 36* (2017), 108–122. 15, 17

[BPA16] BEACCO A., PELECHANO N., ANDÚJAR C.: A survey of real-time crowd rendering. *Comput. Graph. Forum 35* (2016), 32–50. 2

[CBM16] CURTIS S., BEST A., MANOCHA D.: Menge: A modular framework for simulating crowd movement. *Collective Dynamics 1* (2016), 1–40. 18

[CC14] CHARALAMBOUS P., CHRYSANTHOU Y.: The PAG crowd: A graph based approach for efficient data-driven crowd simulation. *Comput. Graph. Forum 33*, 8 (2014), 95–108. 11

[CHD14] CAMPANELLA M. C., HOOGENDOORN S. P., DAAMEN W.: The Nomad model: theory, developments and applications. In *Proc. Conf. Pedestrian and Evacuation Dynamics* (2014), pp. 462–467. 18

[CKGC14] CHARALAMBOUS P., KARAMOUZAS I., GUY S. J., CHRYSANTHOU Y.: A data-driven framework for visual crowd analysis. *Comput. Graph. Forum 33*, 7 (2014), 41–50. 19

[CLEH17] CHEN Y. F., LIU M., EVERETT M., HOW J. P.: Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. In *Proc. IEEE Int. Conf. Robotics and Automation* (2017), pp. 285–292. 13

[CP15] CASADIEGO L., PELECHANO N.: From one to many: Simulating groups of agents with reinforcement learning controllers. In *Proc. Int. Conf. Intelligent Virtual Agents* (2015), pp. 119–123. 12

[DDH13] DUIVES D. C., DAAMEN W., HOOGENDOORN S. P.: State-of-the-art crowd motion simulation models. *Transportation Research Part C: Emerging Technologies 37* (2013), 193–209. 2

[DGAB15] DURUPINAR F., GÜDÜKBAY U., AMAN A., BADLER N. I.: Psychological parameters for crowd simulation: From audiences to mobs. *IEEE Trans. Vis. Comput. Graphics 22*, 9 (2015), 2145–2159. 20

[dLBRM*12] DE LIMA BICHO A., RODRIGUES R. A., MUSSE S. R., JUNG C. R., PARAVISI M., MAGALHÃES L. P.: Simulating crowds based on a space colonization algorithm. *Computers & Graphics 36* (2012), 70–79. 16

[DMCN*17] DUTRA T. B., MARQUES R., CAVALCANTE-NETO J. B., VIDAL C. A., PETTRÉ J.: Gradient-based steering for vision-based crowd simulation algorithms. *Comput. Graph. Forum 36*, 2 (2017), 337–348. 10, 14

[FW07] FAJEN B. R., WARREN W. H.: Behavioral dynamics of intercepting a moving target. *Experimental Brain Research 180*, 2 (2007), 303–319. 9

[GCC*10] GUY S. J., CHHUGANI J., CURTIS S., DUBEY P., LIN M. C., MANOCHA D.: PLEdestrians: a least-effort approach to crowd simulation. In *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation* (2010), Eurographics Association, pp. 119–128. 8, 9, 19

[GF17] GRANT J. M., FLYNN P. J.: Crowd scene understanding from video: A survey. *ACM Trans. Multimedia Comput. Commun. Appl. 13*, 2 (2017). 2, 21

[GJF*18] GUPTA A., JOHNSON J., FEI-FEI L., SAVARESE S., ALAHI A.: Social GAN: Socially acceptable trajectories with generative adversarial networks. In *2018 IEEE/CVF Conf. Computer Vision and Pattern Recognition* (2018), pp. 2255–2264. 12

[GKGG14] GODOY J., KARAMOUZAS I., GUY S. J., GINI M.: Anytime navigation with progressive hindsight optimization. In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems* (2014), pp. 730–735. 15

[GKLM11] GUY S. J., KIM S., LIN M. C., MANOCHA D.: Simulating heterogeneous crowd behaviors using personality trait theory. In *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation* (2011), ACM, pp. 43–52. 8, 20

[GNCL13] GOLAS A., NARAIN R., CURTIS S., LIN M. C.: Hybrid long-range collision avoidance for crowd simulation. *IEEE Trans. Vis. Comput. Graphics 20*, 7 (2013), 1022–1034. 14

[GVDBL*12] GUY S. J., VAN DEN BERG J., LIU W., LAU R., LIN M. C., MANOCHA D.: A statistical similarity measure for aggregate crowd dynamics. *ACM Trans. Graph. 31*, 6 (2012), 1–11. 19

[HBK*10] HÖCKER M., BERKHAHN V., KNEIDL A., BORRMANN A., KLEIN W.: Graph-based approaches for simulating pedestrian dynamics in building models. In *eWork and eBusiness in Architecture, Engineering and Construction* (2010), pp. 389–394. 16, 17

[HBM*20] HAWORTH B., BERSETH G., MOON S., FALOUTSOS P., KAPADIA M.: Deep integration of physical humanoid control and crowd navigation. In *Proc. 13th ACM SIGGRAPH Conf. Motion, Interaction and Games* (2020). 3, 13

[HFV00] HELBING D., FARKAS I., VICSEK T.: Simulating dynamical features of escape panic. *Nature 407* (2000), 487–490. 8

[HKHE12] HELIÖVAARA S., KORHONEN T., HOSTIKKA S., EHTAMO H.: Counterflow model for agent-based simulation of crowd dynamics. *Building and Environment 48* (2012), 89–100. 16

[HM95] HELBING D., MOLNÁR P.: Social force model for pedestrian dynamics. *Physical Review E 51*, 5 (1995), 4282–4286. 5, 6

[HOKP16] HOYET L., OLIVIER A.-H., KULPA R., PETTRÉ J.: Perceptual effect of shoulder motions on crowd animations. *ACM Trans. Graph. 35*, 4 (2016), 1–10. 20

[HPNM16] HE L., PAN J., NARANG S., MANOCHA D.: Dynamic group behaviors for interactive crowd simulation. In *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation* (2016), pp. 139–147. 14

[HXZW20] HE F., XIANG Y., ZHAO X., WANG H.: Informative scene decomposition for crowd analysis, comparison and simulation guidance. *ACM Trans. Graph.* (2020). 19

[JCG13] JAKLIN N. S., COOK IV A. F., GERAERTS R.: Real-time path planning in heterogeneous environments. *Computer Animation and Virtual Worlds 24*, 3 (2013), 285–295. 15

[KBB16] KIELAR P., BIEDERMANN D., BORRMANN A.: *MomenTUMv2: A Modular, Extensible, and Generic Agent-Based Pedestrian Behavior Simulation Framework*. Tech. Rep. TUM-I1643, Technische Universität München, Institut Für Informatik, 2016. 18

[KBG*13] KAPADIA M., BEACCO A., GARCIA F., REDDY V., PELECHANO N., BADLER N. I.: Multi-domain real-time planning in dynamic environments. In *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation* (2013), pp. 115–124. 17

[KGML12] KIM S., GUY S. J., MANOCHA D., LIN M. C.: Interactive simulation of dynamic crowd behaviors using general adaptation syndrome theory. In *Proc. ACM SIGGRAPH Symp. Interactive 3D Graphics and Games* (2012), ACM, pp. 55–62. 20

[KGO09] KARAMOUZAS I., GERAERTS R., OVERMARS M.: Indicative routes for path planning and crowd simulation. In *Proc. 4th Int. Conf. Foundations of Digital Games* (2009), pp. 113–120. 15

[KGvdS12] KARAMOUZAS I., GERAERTS R., VAN DER STAPPEN A. F.: Spacetime group motion planning. In *Proc. 10th Int. Workshop on the Algorithmic Foundations of Robotics* (2012), pp. 227–243. 17

[KHvBO09] KARAMOUZAS I., HEIL P., VAN BEEK P., OVERMARS M.: A predictive collision avoidance model for pedestrian simulation. In *Proc. 2nd Int. Workshop on Motion in Games* (2009), pp. 41–52. 5, 6

[KJG16] KREMYZAS A., JAKLIN N. S., GERAERTS R.: Towards social behavior in virtual-agent navigation. *Science China - Information Sciences 59*, 11 (2016), 112102. 14

[KO10] KARAMOUZAS I., OVERMARS M. H.: A velocity-based approach for simulating human collision avoidance. In *Proc. 10th Int. Conf. Intelligent Virtual Agents* (2010), pp. 180–186. 7, 8, 9

[KO12] KARAMOUZAS I., OVERMARS M. H.: Simulating and evaluating the local behavior of small pedestrian groups. *IEEE Trans. Vis. Comput. Graphics 13* (2012), 394–406. 13

[KPAB15] KAPADIA M., PELECHANO N., ALLBECK J., BADLER N. I.: *Virtual Crowds: Steps Toward Behavioral Realism*. Morgan & Claypool Publishers, 2015. 2

[KSG14] KARAMOUZAS I., SKINNER B., GUY S. J.: Universal power law governing pedestrian interactions. *Phys. Rev. Lett. 113* (2014), 238701:1–5. 6

[KSHF09] KAPADIA M., SINGH S., HEWLETT W., FALOUTSOS P.: Egocentric affordance fields in pedestrian steering. In *Proc. ACM SIGGRAPH Symp. Interactive 3D Graphics and Games* (2009), pp. 215–223. 15

[KSHG18] KARAMOUZAS I., SOHRE N., HU R., GUY S. J.: Crowd space: A predictive crowd analysis technique. *ACM Trans. Graph. 37*, 6 (2018). 19, 21

[KSNG17] KARAMOUZAS I., SOHRE N., NARAIN R., GUY S. J.: Implicit crowds: Optimization integrator for robust crowd simulation. *ACM Trans. Graph. 36*, 4 (2017), 136. 4

[KWS*11] KAPADIA M., WANG M., SINGH S., REINMAN G., FALOUTSOS P.: Scenario space: characterizing coverage, quality, and failure of steering algorithms. In *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation* (2011), pp. 53–62. 19

[KZGK19] KLEINMEIER B., ZÖNNCHEN B., GÖDEL M., KÖSTER G.: Vadere: An open-source simulation framework to promote interdisciplinary understanding. *Collective Dynamics 4* (2019), 1–34. 18

[LA15] LEMERCIER S., AUBERLET J.-M.: Towards more behaviours in crowd simulation. *Computer Animation and Virtual Worlds 27*, 1 (2015), 24–34. 14

[LCHL07] LEE K. H., CHOI M. G., HONG Q., LEE J.: Group behavior from video: A data-driven approach to crowd simulation. In *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation* (2007), pp. 109––118. 11

[LCL07] LERNER A., CHRYSANTHOU Y., LISCHINSKI D.: Crowds by example. *Comput. Graph. Forum 26*, 3 (2007), 655–664. 11, 19

[LCMP19a] LÓPEZ A., CHAUMETTE F., MARCHAND E., PETTRÉ J.: Attracted by light: vision-based steering virtual characters among dark and light obstacles. In *Proc. 12th ACM SIGGRAPH Conf. Motion, Interaction and Games* (2019). 10

[LCMP19b] LÓPEZ A., CHAUMETTE F., MARCHAND E., PETTRÉ J.: Character navigation in dynamic environments based on optical flow. *Comput. Graph. Forum 38*, 2 (2019), 181–192. 10, 14

[LCSCO09] LERNER A., CHRYSANTHOU Y., SHAMIR A., COHEN-OR D.: Data driven evaluation of crowds. In *Proc. Int. Workshop on Motion in Games* (2009), Springer, pp. 75–83. 19

[LCW*15] LI T., CHANG H., WANG M., NI B., HONG R., YAN S.: Crowded scene analysis: A survey. *IEEE Trans. Circuits Sys!. Video Technol. 25*, 3 (2015), 367–386. 2, 21

[LJK*12] LEMERCIER S., JELIC A., KULPA R., HUA J., FEHRENBACH J., DEGOND P., APPERT-ROLLAND C., DONIKIAN S., PETTRÉ J.: Realistic following behaviors for crowd simulation. *Comput. Graph. Forum 31*, 2 (2012), 489–498. 14

[LLL12] LOPEZ T., LAMARCHE F., LI T.-Y.: Space-time planning in changing environments: using dynamic objects for accessibility. *Computer Animation and Virtual Worlds 23* (2012), 87–99. 17

[LWL18] LEE J., WON J., LEE J.: Crowd simulation by deep reinforcement learning. In *Proc. 11th ACM SIGGRAPH Conf. Motion, Interaction and Games* (2018). 13

[MBV*18] MEERHOFF L., BRUNEAU J., VU A., OLIVIER A.-H., PETTRÉ J.: Guided by gaze: prioritization strategy when navigating through a virtual crowd can be assessed through gaze activity. *Acta Psychologica 190* (2018), 248—257. 21

[MHT11] MOUSSAÏD M., HELBING D., THERAULAZ G.: How simple rules determine pedestrian behavior and crowd disasters. *Proc. National Academy of Sciences 108* (2011), 6884–6888. 7, 8, 9, 10

[MK19] MAVROGIANNIS C. I., KNEPPER R. A.: Multi-agent path topology in support of socially competent navigation planning. *International Journal of Robotics Research 38*, 2–3 (2019), 338–356. 15, 17

[MKT*16] MOUSSAÏD M., KAPADIA M., THRASH T., SUMNER R. W., GROSS M., HELBING D., HÖLSCHER C.: Crowd behaviour during high-stress evacuations in an immersive virtual environment. *Journal of The Royal Society Interface 13*, 122 (2016), 20160414. 21

[MLC*08] MCDONNELL R., LARKIN M., COLLINS S., O'SULLIVAN C., ET AL.: Clone attack! perception of crowd variety. *ACM Trans. Graph. 27*, 3 (2008). 20

[MMON10] MCHUGH J. E., MCDONNELL R., O'SULLIVAN C., NEWELL F. N.: Perceiving emotion in crowds: the role of dynamic body postures on the perception of emotion in crowded scenes. *Experimental brain research 204*, 3 (2010), 361–372. 20

[MPG*10] MOUSSAÏD M., PEROZO N., GARNIER S., HELBING D., THERAULAZ G.: The walking behaviour of pedestrian social groups and its impact on crowd dynamics. *PloS one 5*, 4 (2010), e10047. 13

[MTK18] MAVROGIANNIS C. I., THOMASON W. B., KNEPPER R. A.: Social momentum: A framework for legible navigation in dynamic multi-agent environments. In *Proc. 2018 ACM/IEEE Int. Conf. Human-Robot Interaction* (2018), pp. 361—-369. 15

[NBM17] NARANG S., BEST A., MANOCHA D.: Interactive simulation of local interactions in dense crowds using elliptical agents. *Journal of Statistical Mechanics: Theory and Experiment 2017*, 3 (2017), 033403. 8

[OPOD10] ONDŘEJ J., PETTRÉ J., OLIVIER A.-H., DONIKIAN S.: A synthetic-vision based steering approach for crowd simulation. *ACM Trans. Graph. 29*, 4 (2010), 123. 9, 10

[PAKB16] PELECHANO N., ALLBECK J. M., KAPADIA M., BADLER N. I.: *Simulating Heterogeneous Crowds with Interactive Behaviors*. CRC Press, 2016. 2

[POO*09] PETTRÉ J., ONDŘEJ J., OLIVIER A.-H., CRETUAL A., DONIKIAN S.: Experiment-based modeling, simulation and validation of interactions between virtual walkers. In *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation* (2009), pp. 189–198. 19

[PPD07] PARIS S., PETTRÉ J., DONIKIAN S.: Pedestrian reactive navigation for crowd simulation: a predictive approach. *Comput. Graph. Forum 26*, 3 (2007), 665–674. 7

[PRY13] PARK J. H., ROJAS F. A., YANG H. S.: A collision avoidance behavior model for crowd simulation based on psychological findings. *Computer Animation and Virtual Worlds 24* (2013), 173–183. 9

[PSAB07] PELECHANO N., STOCKER C., ALLBECK J., BADLER N.: Feeling crowded? exploring presence in virtual crowds. In *Proc. 10th Int. Workshop on Presence* (2007), pp. 373–376. 21

[QH10] QIU F., HU X.: Modeling group structures in pedestrian crowd simulation. *Simulation Modelling Practice and Theory 18*, 2 (2010), 190–205. 13

[QZK*19] QIAO G., ZHOU H., KAPADIA M., YOON S., PAVLOVIC V.: Scenario generalization of data-driven imitation models in crowd simulation. In *Proc. 12th ACM SIGGRAPH Conf. Motion, Interaction and Games* (2019), pp. 36:1–36:11. 20

[RCB*17] REN Z., CHARALAMBOUS P., BRUNEAU J., PENG Q., PETTRÉ J.: Group modeling: A unified velocity-based approach. *Comput. Graph. Forum 36*, 8 (2017), 45–56. 13

[Rey87] REYNOLDS C. W.: Flocks, herds and schools: A distributed behavioral model. In *Proc. 14th Conf. Computer graphics and interactive techniques* (1987), pp. 25–34. 13

[Rey99] REYNOLDS C. W.: Steering behaviors for autonomous characters. In *Game developers conference* (1999), vol. 1999, Citeseer, pp. 763–782. 13, 15

[RPH*20] RUDENKO A., PALMIERI L., HERMAN M., KITANI K. M., GAVRILA D. M., ARRAS K. O.: Human motion trajectory prediction: a survey. *International Journal of Robotics Research 39*, 8 (2020), 895–935. 12, 21

[RRW14] RIO K. W., RHEA C. K., WARREN W. H.: Follow the leader: Visual control of speed in pedestrian following. *Journal of Vision 14*, 2 (2014), 4:1–4:16. 14

[RTT90] RENAULT O., THALMANN N. M., THALMANN D.: A vision-based approach to behavioural animation. *The Journal of Visualization and Computer Animation 1*, 1 (1990), 18–21. 9

[RXX*21] REN J., XIANG W., XIAO Y., YANG R., MANOCHA D., JIN X.: Heter-sim: Heterogeneous multi-agent systems simulation by interactive data-driven optimization. *IEEE Trans. Vis. Comput. Graphics 27*, 3 (2021), 1953–1966. Pre-print available online since 2018. 12

[SKFR09] SINGH S., KAPADIA M., FALOUTSOS P., REINMAN G.: SteerBench: a benchmark suite for evaluating steering behaviors. *Computer Animation and Virtual Worlds 20*, 5–6 (2009), 533–548. 18, 19

[SKH*11] SINGH S., KAPADIA M., HEWLETT B., REINMAN G., FALOUTSOS P.: A modular framework for adaptive agent-based steering. In *Proc. ACM SIGGRAPH Symp. Interactive 3D Graphics and Games* (2011), pp. 141–150. 17, 18

[SKRF11] SINGH S., KAPADIA M., REINMAN G., FALOUTSOS P.: Footstep navigation for dynamic crowds. *Computer Animation and Virtual Worlds 22*, 2–3 (2011), 151—-158. 3, 15

[SMKB13] SHOULSON A., MARSHAK N., KAPADIA M., BADLER N. I.: ADAPT: The agent development and prototyping testbed. In *Proc. 17th ACM SIGGRAPH Symp. Interactive 3D Graphics and Games* (2013), pp. 9–18. 18

[SMTTvdS16] STÜVEL S. A., MAGNENAT-THALMANN N., THALMANN D., VAN DER STAPPEN A. F.: Torso crowds. *IEEE Trans. Vis. Comput. Graphics 23*, 7 (2016), 1823–1837. 3, 15

[ST07] SHAO W., TERZOPOULOS D.: Autonomous pedestrians. *Graphical Models 69* (2007), 246–274. 16

[TM13] THALMANN D., MUSSE S. R.: *Crowd Simulation*, 2nd ed. Springer, 2013. 2

[vdBGLM11] VAN DEN BERG J. P., GUY S. J., LIN M. C., MANOCHA D.: Reciprocal n-body collision avoidance. In *Proc. 14th Int. Symp. Robotics Research* (2011), pp. 3–19. 6, 8, 9, 11

[vdBLM08] VAN DEN BERG J., LIN M., MANOCHA D.: Reciprocal velocity obstacles for real-time multi-agent navigation. In *Proc. IEEE Int. Conf. Robotics and Automation* (2008), IEEE, pp. 1928–1935. 6, 7

[vdBSGM11] VAN DEN BERG J. P., SNAPE J., GUY S. J., MANOCHA D.: Reciprocal collision avoidance with acceleration-velocity obstacles. In *Proc. IEEE Int. Conf. Robotics and Automation* (2011), pp. 3475–3482. 8, 13

[vGJCG15] VAN GOETHEM A., JAKLIN N. S., COOK IV A. F., GERAERTS R.: On streams and incentives: A synthesis of individual and collective crowd motion. In *Proc. 28th Int. Conf. Computer Animation and Social Agents* (2015), pp. 29–32. 16

[vTCG12] VAN TOLL W. G., COOK IV A. F., GERAERTS R.: Real-time density-based crowd simulation. *Computer Animation and Virtual Worlds 23*, 1 (2012), 59–69. 17

[vTGL*20] VAN TOLL W., GRZESKOWIAK F., LÓPEZ A., AMIRIAN J., BERTON F., BRUNEAU J., DANIEL B. C., JOVANE A., PETTRÉ J.: Generalized microscropic crowd simulation using costs in velocity space. In *Proc. ACM SIGGRAPH Symp. Interactive 3D Graphics and Games* (2020). 4, 10, 18, 21

[vTJG15] VAN TOLL W., JAKLIN N., GERAERTS R.: Towards believable crowds: A generic multi-level framework for agent navigation. In *ASCI.OPEN / ICT.OPEN (ASCI track)* (2015). 18

[vTP19] VAN TOLL W., PETTRÉ J.: Connecting global and local agent navigation via topology. In *Proc. 12th ACM SIGGRAPH Conf. Motion, Interaction and Games* (2019). 17

[vTP20] VAN TOLL W., PETTRÉ J.: Synchronizing navigation algorithms for crowd simulation via topological strategies. *Computers & Graphics 89* (2020), 24–37. 17

[War06] WARREN W. H.: The dynamics of perception and action. *Psychological Review 113*, 2 (2006), 358. 9

[War18] WARREN W. H.: Collective motion in human crowds. *Current Directions in Psychological Science 27*, 4 (2018), 232–240. 14

[Wei93] WEIDMANN U.: Transporttechnik der Fussgänger - Transporttechnische Eigenschaften des Fussgängerverkehrs. Literature Research 90, ETH Zürich, Institut für Verkehrsplanung, Transporttechnik, Strassen- und Eisenbahnbau, 1993. In German. 18

[WF08] WARREN W. H., FAJEN B. R.: Behavioral dynamics of visually-guided locomotion. In *Coordination: Neural, behavioral and social dynamics*. Springer, 2008, pp. 45–75. 9

[WGO*14] WOLINSKI D., GUY S. J., OLIVIER A.-H., LIN M., MANOCHA D., PETTRÉ J.: Parameter estimation and comparative evaluation of crowd simulations. *Comput. Graph. Forum 33*, 2 (2014), 303–312. 19

[WJLT17] WEISS T., JIANG C., LITTENEKER A., TERZOPOULOS D.: Position-based multi-agent dynamics for real-time crowd simulation. In *Proc. 10th ACM SIGGRAPH Int. Conf. Motion in Games* (2017), pp. 10:1–10:8. 4

[WLP16] WOLINSKI D., LIN M. C., PETTRÉ J.: WarpDriver: Context-aware probabilistic motion prediction for crowd simulation. *ACM Trans. Graph. 35*, 6 (2016), 164:1–164:11. 15

[WOO17] WANG H., ONDŘEJ J., O'SULLIVAN C.: Trending paths: A new semantic-level metric for comparing simulated and real crowd data. *IEEE Trans. Vis. Comput. Graphics 23*, 5 (2017), 1454–1464. 19

[XJJD14] XU M.-L., JIANG H., JIN X.-G., DENG Z.: Crowd simulation and its applications: recent advances. *Journal of Computer Science and Technology 29* (2014), 799–811. 2

[YLG*20] YANG S., LI T., GONG X., PENG B., HU J.: A review on crowd simulation and modeling. *Graphical Models 111* (2020), 101081. 2

[ZIK11] ZANLUNGO F., IKEDA T., KANDA T.: Social force model with explicit collision prediction. *EPL (Europhysics Letters) 93*, 6 (2011), 68005. 5, 6

[ZTC13] ZHAO M., TURNER S. J., CAI W.: A data-driven crowd simulation model based on clustering and classification. In *Proc. IEEE/ACM 17th Int. Symp. Distributed Simulation and Real Time Applications* (2013), pp. 125–134. 11