

Group Modeling: A Unified Velocity-Based Approach

Z Ren, Panayiotis Charalambous, Julien Bruneau, Qunsheng Peng, Julien Pettré

▶ **To cite this version:**

Z Ren, Panayiotis Charalambous, Julien Bruneau, Qunsheng Peng, Julien Pettré. Group Modeling: A Unified Velocity-Based Approach. Computer Graphics Forum, 2017, 36 (8), pp.45-56. 10.1111/cgf.12993 . hal-01372766

HAL Id: hal-01372766
https://inria.hal.science/hal-01372766

Submitted on 27 Sep 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Volume xx (200y), Number z, pp. 1–11

Group Modeling: a Unified Velocity-based Approach

Z. Ren[1†] P. Charalambous[2‡] J. Bruneau[3§] Q. Peng[1] and J. Pettré[2¶]

[1]Zhejiang University, Hangzhou, China [2]Inria Rennes, France [3]IRISA, University Rennes 1, France



(a) Performance (b) Pedestrians (c) Tourists/Guide (d) Teachers/Students

Figure 1: Our simulation algorithm is capable of generating different group behavior using a small set of intuitive parameters.

Abstract

Crowd simulators are commonly used to populate movie or game scenes in the entertainment industry. Even though it is crucial to consider the presence of groups for the believability of a virtual crowd, most crowd simulations only take into account individual characters or a limited set of group behaviors. We introduce a unified solution that allows for simulations of crowds that have diverse group properties such as social groups, marches, tourists and guides, etc. We extend the Velocity Obstacle approach for agent based crowd simulations by introducing Velocity Connection; the set of velocities that keep agents moving together whilst avoiding collisions and achieving goals. We demonstrate our approach to be robust, controllable, and able to cover a large set of group behaviors.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation; I.6.8 [Simulation and Modeling]: Types of Simulation—Animation

1. Introduction

Real crowds are not only made of isolated individuals but also structured by the presence of groups, such as families, bands of friends, larger groups of schoolmates or tourists during a field-trip. Groups result into visually strong patterns: aggregations, specific spatial distributions, cohesive motions, etc. In many places, crowds are mainly composed of groups compared to individual walkers (e.g., 70% in the example of a shopping street according to Moussaïd et al. [MPG*10]). When aiming to improve crowd simulator quality, it is crucial to consider groups, especially in the context of entertainment applications which are in demand for high visual quality. The general objective of our work is to simulate crowds that may contain groups.

While several techniques that simulate groups in crowds are reported in the literature, the problem has only received partial answers: solutions are limited to specific situations, group types or behaviors. We first address the problem of providing a general definition of groups in crowds which may exist under many forms. Groups are of different **sizes**: a family of four, ten friends, fifty tourists, one thousand in a march, etc. Groups are structured by different internal **relations**: followers with a leader, tourists made of families with a guide, schoolmates with a set of teachers, etc. Groups are guided by different **goals** and **constraints**: armies have specific formations, parents keep contact with their kids, friends chat together, etc. It is still a challenge to model groups in all these dimensions. Previous solutions *do not generalize to a large variety of groups*. We also address the problem of *implementing* groups in a crowd simulator. Previous approaches added an upper simula-

† e-mail: renzhiguo@outlook.com
‡ e-mail: panayiotis.charalambous@inria.fr
§ e-mail: julien.bruneau@irisa.fr
¶ e-mail: julien.pettre@inria.fr

tion behavioral layer to play on agents' goals and force grouping. This a limited solution, we believe that the notion of groups should be expressed at the core of microscopic simulation algorithms, as a new mode of local interactions between agents. This requires to mathematically define those local interactions and to deal with the problem of combining several, often conflicting, local interactions (e.g., avoiding collisions while staying close to other agents).

In this paper, we propose a unified and opened definition of groups in crowds: a group is a subset of agents which desire to move together. This means that agents of the same group should move so that they maintain a bounded distance. Setting both upper and lower bounds allows agents to simultaneously group and avoid collisions respectively. We implement this definition based on the velocity-obstacle formalism. One agent in a group constantly moves with velocities that enable: a) not colliding with other neighbor agents, and b) staying at a bounded distance to some other agents belonging in the same group. On the technical point of view, we extend the RVO algorithm by adding an upper bound on the future distance of closest approach to make agents group together. Obviously, in most cases, an agent cannot stay at a bounded distance from all the other members of its group especially when that group is large. Therefore, one important component of our solution is the way each agent selects a subset of neighbors to stay close to (the *connection-agents*). Playing on the few parameters of our algorithm (the number of connection agents, their selection, the bound distance value with connection agents, etc.) we demonstrate the remarkable ability of our algorithm to generate a large variety of emerging group behaviors as well as to deal with complex simulation scenarios.

Our contribution is multi-faceted:

1. We introduce a new model of local interaction for grouping, based on a velocity-based formulation.
2. We propose a unified group simulation algorithm which is capable of covering a large set of existing situations. By only playing on few parameters' value, we are able to configure our algorithm to consider small and large groups, uniform or leader/followers groups, cohesive groups or sparse ones, etc.
3. Our method enables emergent group behaviors based on the local expression of grouping interactions. We demonstrate the variety of emergent behaviors at the same time we map them with the simulation parameters values.

The remaining of the paper is organized as follows. In Section 2, we overview existing solutions to simulate groups and outline the lack of a general solution to simulate them. Section 3 is dedicated to the technical description of our approach. In Section 4 we demonstrate our approach in two different ways. We first demonstrate the variety of group behaviors emerging from different parameter sets. Secondly, we demonstrate our method over typical situations of real usage. We discuss our results and the limitations of our approach before concluding and opening perspectives for future research.

2. Related Work

This section overviews the literature on crowd simulation and focuses on group modeling; we do not report the sociological aspects of the topic.

Microscopic crowd simulation Modeling local interactions is an essential task in the design of microscopic crowd simulators. Many approaches addressed the problem of collision avoidance [Rey99, HM95, POO*09, GCC*10, OPOD10], respectively based on steering rules, social forces, experimental analysis, velocity space optimization and vision perception. Our method is built on the velocity obstacle (VO) concept [FS98]. Van den Berg et al. [VdBLM08] proposed the Reciprocal Velocity Obstacle (RVO) to handle oscillations present in the initial VO method, and later derived a linear programming solution [VDBGLM11]. Guy et al. [GKLM11] mapped RVO parameters with personality descriptors. Kim et al. [KGL*13] learned simulation parameters from video. These only consider the collision avoidance between individuals, other techniques enable individuals to avoid a group. Schuerman et al. [SSKF10] represented the whole group as a new type of agent. Lemercier and Auberlet [LA15] introduced behavioral laws to simulate group avoidance. Bruneau et al. [BOP15] used Virtual Reality to study how real humans behave when avoiding groups.

Group simulation The seminal work by Reynolds [Rey87] demonstrated impressive flocking simulation. Musse and Thalmann [MT97] considered the relationship between groups of individuals and the emergent group behavior originating from it. The same authors presented a hierarchical model which can control groups with different degrees of autonomy [MT01]. Helbing's social forces model [HM95] was extended to simulate groups by adding several attractive forces [PV08, BMdOB03]. The leader-follower group structure has been widely considered. Loscos et al. [LMM03] presented a model in which the leader decides about the motion of the entire group, guided by local rules. Qiu et al. [QH10] algebraically modeled inter-group and intra-group relations. Villamil et al. [VMdO03] discussed relations between agents' internal states and the ability of grouping.

Small-group simulation has attracted more interest: some empirical studies regarding the spatial organization of pedestrian groups [PE09, MPG*10] have been conducted. Based on observations, Moussaïd et al. employed a social force model-based approach to organize group formations, while Peters and Ennis used discrete formation templates, as Karamouzas and Overmars [KO12]. Yeh et al. [YCP*08] placed different types of composite agents to constrain agents in groups. Rojas and Yang [RY13] introduced a group agent consisting of several hinge-connected slots which each member follows. Alonso-Mora et al. [AMBR*12] enabled a large number of robots to transform among a set of formations by computing their goal positions in each formation assignment. In the robotics community, Kimmel et al. [KDB12] maintain cohesion for a group of robots based on the VO approach. Besides the microscopic simulation methods mentioned above, macroscopic simulations can display macroscopic behavior of the whole crowd based on continuum or fluid dynamics [TCP06, NGCL09]. The computer animation community also values the user interaction of the group control in order to design a required movement [KLLT08, TYK*09]. Recently, example-based and data-driven approaches have also been used to construct groups according to data from videos of real crowds [LCL07, CC14, JCP*10, LCHL07].

Analysis Crowd simulation has been very active recently and

many different models have been proposed to simulate them; most of them are designed to first consider crowds made of individual agents. It is only in a second step that they are then extended to consider moving groups [SSKF10, LA15, BOP15]. However, those extensions generally focus on one specific aspect of group behaviors, like flocking, leading and following, moving in formations, etc. No simulation methods have given the general formulation of what is a group. As a result, methods that suit small group simulation will not suit larger groups, or flocking algorithms won't handle leader-followers situations. Although rule-based and force-based methods can intuitively model group interactions, the quality of the simulated result has a high dependence on the complexity of rules and the definition of force components.

In comparison to previous work, we attempt to generate a group in a general bottom-up way; i.e., the group and its pattern are emerged and evolved due to the interactions between individuals and different behaviors can be achieved by tuning parameters of individuals without the need of group-level rules.

Our interaction algorithm is based on the concept of VO, which can easily be implemented and is computational efficient so that it is suitable for real-time applications. We extend the concept of VO, so that the grouping phenomenon can be reproduced directly through the interactions between individuals. This approach is in contrast with previous attempts also based on VO, such as Kimmel et al. [KDB12]. In this latter approach, group structure is fixed, agents have the same speed and velocity is defined by direction and it cannot handle various situations like a leader-follower group. In comparison, our neighbor selection mechanism enables emergent group splitting and merging under external constraints, and the way we define internal relations between members enables handling a larger set of group behaviors and situations.

3. Velocity-based group simulation algorithm

3.1. Overview

Our solution simulates crowds made of both individuals and groups based on a velocity-obstacle approach. *Individual agents* are agents that navigate with no desire to stay close to another agent and are steered exactly like in the existing RVO algorithm [VdBLM08]: their velocity is so that their (short term) future distance of closest approach with any other agent is always above a collision threshold. *Group agents* have a desire to stay near specific agents (the group members); these agents combine local avoidance with grouping. Grouping is performed by selecting at simulation time a subset of agents belonging to the same group, called the *connection neighbors*. A group agent's velocity is then computed so that its short term future distance of closest approach with its connection neighbors is below a maximum distance threshold. More precisely, our method is based on the run loop illustrated in Figure 2. During preliminary initialization, a user sets relationships between agents (Section 3.2). Then, during simulation, each agent: 1) selects a subset of connection-neighbors from its group (Section 3.3); 2) computes the set of velocities that allow it to move towards its destination while avoiding collisions and staying near its connection neighbors (Section 3.4); 3) selects a new velocity from this set (Section 3.5) and finally 4) applies the selected velocity and moves.

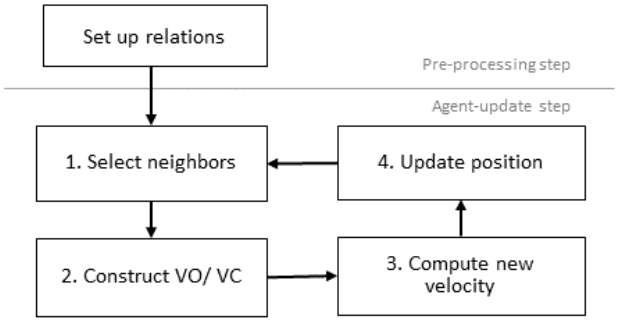


Figure 2: Overview of Simulation workflow.

3.2. Setting up Relations

Users initialize a simulation by setting relation influences between agents and thus creating group of agents that are related. This step is performed through a *relation matrix* that encodes group relationships between agents. Entry $e_{ij} \in [0,1]$ of the matrix represents the relation influence, i.e., the preference of agent A_i to stay close to A_j: high values indicate high desire whereas a value of 0 indicates no grouping relationship. Conceptually, the relation matrix is a *weighted directed graph* describing logical (often asymmetric) relationships between agents. We can see a simple illustrative example of a relation matrix in Table 1: a crowd of 5 agents consisting of 2 groups: $\{A_1, A_2, A_3\}$ and $\{A_4, A_5\}$. In the first group, A_1 has a stronger desire to stay near A_3 than to A_2 ($e_{13} > e_{12}$).

	A_1	A_2	A_3	A_4	A_5
A_1	-	.7	1	0	0
A_2	.7	-	1	0	0
A_3	1	1	-	0	0
A_4	0	0	0	-	1
A_5	0	0	0	1	-

Table 1: Example of a relation matrix for two groups.

3.3. Selecting Connection-Neighbors

At runtime, each agent belonging in a group moves to stay close to some of its influencing agents (according to the relation matrix).The number of followed agents is limited, because maintaining close distance to all can be infeasible. As demonstrated in Section 4, the limit number of connection-neighbors n^c is a main parameter of our method that affects emergent group behavior. This section describes how we select for each group agent a subset of connection agents, among all its influencing agents.

Selection Criteria We define a *relation distance* d_{ij}^{rel} between agents A_i and A_j that takes into account both the euclidean distance d_{ij}^{euc} and relation influence e_{ij}:

$$d_{ij}^{rel} = d_{ij}^{euc} \cdot f(e_{ij}), \tag{1}$$

where $f(.)$ is a monotonically decreasing function; i.e., $f(0) \geq$

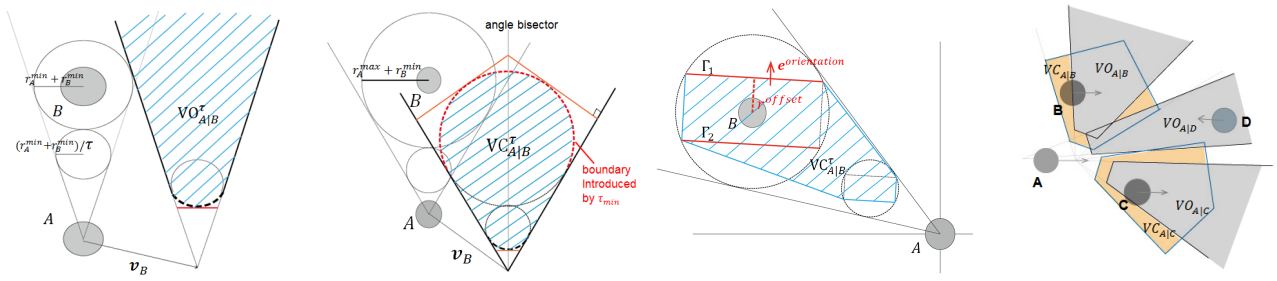


Figure 3: (a) The velocity obstacle $VO^{\tau}_{A|B}$ of a agent B to agent A is a truncated cone in the velocity space. (b) The velocity connection $VC^{\tau}_{A|B}$ of an agent B to agent A contains the velocities that result in grouping A and B together for at least τ_{min} seconds before time τ. Here $\tau_{min} = 1$. (c) $VC^{\tau}_{A|B}$ for a rectangular constraint. (d) A situation where not all velocity constraints can be satisfied since $VC^{\tau}_{A|B} \cap VC^{\tau}_{A|C} = \emptyset$ (group neighbors B and C are not close to each other). In this case, agent A tries to stay close to both B and C.

$f(e_{ij}) \geq f(1), \forall e_{ij} \in [0,1]$[†]. Neighbors having the lowest n^c relation distances will be selected as the connection neighbors. Combining euclidean distance and relation influence ensures that nearby low influencing agents will not overshadow further away high influencing agents.

During simulation, when several agents are influencing each other they form a spatially distributed group. As all group agents are selecting a limited number of connection-neighbors (potentially smaller than the total number of agents in the group), a group can split into several subgroups. This happens in an emergent manner, depending on the group constraints. As an example, consider a school trip where all the members of the school are initially together and might split into several subgroups due to navigation purposes or because they have different goals. This emergent behavior is studied in Section 4.1 and discussed in Section 5.

3.4. Velocity Connection

Having the set of connection neighbors for all agents, we need to compute a velocity for each to perform both collision avoidance and grouping. Technically speaking, we build on the principle of reciprocal velocity obstacles (RVO [VdBLM08]). RVO allows performing collision avoidance by always selecting a walking velocity so that the distance of closest approach over a future time window is *above a collision distance threshold* r^{min}. To make an agent move near another agent, we add an upper bound r^{max} to the distance of closest approach. The velocity of the agent will combine two local interactions: collision avoidance and staying close. In other words, it keeps distances with both a minimum and maximum boundary.

Velocity Obstacle We first remind how the velocity obstacle VO set is computed in an agent's reachable velocity space. Let $D(\mathbf{p}, r)$ denote an open disc of radius r centered at $\mathbf{p}$. Given two agents A and B, the velocity obstacle for A induced by B is the set of all velocities of A that will cause a collision between A and B at some moment before time τ. Formally:

$$VO^{\tau}_{A|B} = \{\mathbf{v} | \exists t \in [0,\tau] : t(\mathbf{v} + \mathbf{v}_B) \in D(\mathbf{p}_B - \mathbf{p}_A, r^{min}_A + r^{min}_B)\}. \quad (2)$$

In the velocity space, VO is a truncated cone and τ determines its truncated boundary, as shown in Figure 3a. Velocities outside VO are guaranteed to be collision free, which leads to the definition of the set of collision avoiding (CA) velocities for A:

$$CA^{\tau}_{A|B} = \{\mathbf{v} | \mathbf{v} \notin VO^{\tau}_{A|B}\}. \quad (3)$$

Velocity Connection We define the velocity connection VC as the set of velocities that make the agent's distance to another agent within an upper bound in at latest time τ. For an agent A, the velocity connection for A with respect to B can be formulated in a similar way as VO:

$$VC^{\tau}_{A|B} = \{\mathbf{v} | \exists t \in [\tau_{min},\tau] : t(\mathbf{v} + \mathbf{v_B}) \in D(\mathbf{p}_B - \mathbf{p}_A, r^{max}_A + r^{min}_B)\}, \quad (4)$$

where τ_{min} is the minimum time spent to group together, which should be more than the update period to prevent excessive displacements in one time step. As shown in Figure 3b, τ_{min} adds another boundary to the VC as compared to VO. In our implementation, each of the curved boundaries of VC and VO are replaced by a tangent line for efficiency.

When A succeeds in approaching B at a distance less than r^{max}_A, A should preserve that distance at all times. Thus, the relative velocity should stay in a circle centered at $\mathbf{p}_B - \mathbf{p}_A$ with radius r^{max}_A and thus the velocity connection becomes:

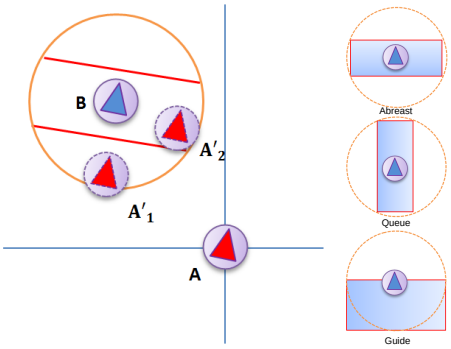
$$VC_{A|B} = \{\mathbf{v} | \tau_{min}(\mathbf{v} + \mathbf{v_B}) \in D(\mathbf{p}_B - \mathbf{p}_A, r^{max}_A + r^{min}_B)\}. \quad (5)$$

Formation Constraints The r^{max}_A boundary means that agents

[†] We use an exponentially decreasing function.

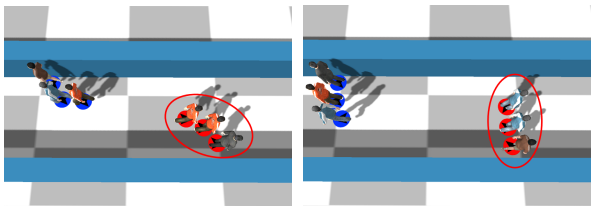
tend to stay in an arbitrary position within a circle around neighbors. However, groups often move according to a specific formation, e.g., members of small groups prefer to walk side by side. Considering this, users can create additional formation constraints. For some situations, we employ rectangular constraints by placing boundaries Γ_1 and Γ_2 as shown in Figure 3c. Placement is determined by an offset r^{offset} and a direction $\mathbf{e}^{orientation}$ depending on requested formations. The VC in then obtained by considering the vertices of the rectangle at the current timestep and after τ seconds.

As an example, consider agent A with group neighbor B (Figure 4a). By setting $\mathbf{e}^{orientation}$ to the moving direction B, A aims in staying in an "abreast" formation with B. In the current implementation, we have three types of rectangular constraints, which can give queuing, abreast and guide-follower patterns (Figure 4b). As we show in Figure 5, agents keep their abreast formation even after interacting in constrained environments. Generally speaking, the constraint area can be any polygonal shape in the velocity space. If the polygon is convex, the grouping velocity set can be obtained by computing the convex hull of the vertices of the constraint polygon and the polygon scaled by τ.



(a) (b)

Figure 4: **Anisotropic Constraints.** (a) A'_1 and A'_2 are possible future positions for agent A. Both of them are acceptable in the isotropic constraint whereas only A'_2 is acceptable when having a rectangular abreast constraint. (b) The currently implemented anisotropic constraints.



(a) Isotropic ([KDB12]) (b) Abreast Formation

Figure 5: (a) Using an isotropic constraint, agents will keep their distances but not initial angles, whereas (b) using an abreast constraint forces agents to adapt as requested.

Multiple VCs and VOs In multi-agent scenes, collision free velocities can be chosen from the intersection of CA's of multiple nearby agents. Velocities keeping agents together to multiple connection neighbors would be in the intersection of VCs. Therefore velocities lying in both of the two intersections satisfy both grouping constraints and collision avoidance. However, as shown in Figure 3d, there are situations that either of the intersections becomes empty. In these cases, rather than selecting a velocity strictly from the intersection set, we consider the preference of all reachable velocities and the most preferred velocity is selected.

3.5. Velocity Preference

During simulation, a group agent has to move towards a target while avoiding collisions and staying close to influencing agents. Therefore, an agent has to select a velocity that balances between these three, often conflicting criteria. We define $m(\mathbf{v})$, as the preference value of selecting velocity $\mathbf{v}$:

$$m(\mathbf{v}) = m^{VO}(\mathbf{v}) + w_g m^{VC}(\mathbf{v}) + w_v m^{des}(\mathbf{v}). \quad (6)$$

Here, m^{VO}, m^{VC} and m^{des} indicate preferences for avoiding collisions, staying with influencing agents (group cohesiveness) and following desired velocity respectively. Each one of these components is weighted by 1, w_g and w_v respectively ($w_g, w_v \leq 1$); i.e., velocities that avoid collisions are typically preferred. High values of $m(\mathbf{v})$ indicate high preference of selecting velocity $\mathbf{v}$. Having a set of candidate velocities $\mathbb{V} = \{\mathbf{v} : |\mathbf{v}| \in D(\mathbf{0}, |\mathbf{v}|^{max})\}$, we set the agent's new velocity $\mathbf{v}^{new}$ as the one with the maximal preference value:

$$\mathbf{v}^{new} = \arg\max_{\mathbf{v}\in\mathbb{V}} m(\mathbf{v}). \quad (7)$$

$|\mathbf{v}|^{max}$ is the maximum speed the agent can reach and is determined by the agent's kinematic limitations and $D(\mathbf{0}, |\mathbf{v}|^{max})$ contains all the reachable velocities of an agent. The candidate velocity set $\mathbb{V}$ is constructed by sampling a number of velocities distributed over $D(\mathbf{0}, |\mathbf{v}|^{max})$. In the following paragraphs, we describe the three components of Equation 6.

Grouping For an agent A_i having a set of connection neighbors $\mathbb{CN}_i$, preference to the velocity connection m^{VC} is dependant on how far away the candidate velocity is from all velocity connections $\{VC_{i|j} : A_j \in \mathbb{CN}_i\}$. We set:

$$m^{VC}(\mathbf{v}) = -\frac{1}{|\mathbb{CN}_i|} \sum_{A_j \in \mathbb{CN}_i} e_{ij} \lambda_j^{VC}(\mathbf{v}), \quad (8)$$

where $|\mathbb{CN}_i|$ is the number of connections neighbors of agent A_i. e_{ij} is the relation influence of agent A_j to agent A_i (Section 3.2); large values of e_{ij} increase the attraction of A_j to A_i so that the pre-

ferred velocity is closer to $VC_{i|j}$. Finally $\lambda_j^{VC}(\mathbf{v})$ weighs velocities based on distance to the boundary of velocity connections:

$$\lambda_j^{VC}(\mathbf{v}) = \begin{cases} d_j^{VC} & \text{if } \mathbf{v} \notin VC_{i|j} \\ 0 & \text{if } \mathbf{v} \in VC_{i|j} \end{cases}. \quad (9)$$

d_j^{VC} is the nearest distance from $\mathbf{v}$ to the boundary of $VC_{i|j}$.

Collision avoidance Differently to m^{VC}, m^{VO} considers the distance from $\mathbf{v}$ to $VO_{i|j}$. Let $\lambda_j^{VO}(\mathbf{v})$ be:

$$\lambda_j^{VO}(\mathbf{v}) = \begin{cases} d_j^{VO} & \text{if } \mathbf{v} \in VO_{i|j} \\ 0 & \text{if } \mathbf{v} \notin VO_{i|j} \end{cases} \quad (10)$$

Here d_j^{VO} is the nearest distance from $\mathbf{v}$ to the boundary of $VO_{i|j}$. The maximum $\lambda_j^{VO}(\mathbf{v})$ measures how much that agent violates the constraints. Thus, we use it to penalize choosing $\mathbf{v}$; $m^{VO}(\mathbf{v})$ is defined as the negative of this penalty:

$$m^{VO}(\mathbf{v}) = -max(\lambda_j^{VO}(\mathbf{v})) \quad (11)$$

Desired velocity Finally, to define a preference to desired velocity m^{des}, we set $\lambda^{des}(\mathbf{v})$ to be the distance between $\mathbf{v}$ and the desired velocity. Thus m^{des} is defined as:

$$m^{des}(\mathbf{v}) = -\lambda^{des}(\mathbf{v}) \quad (12)$$

In Section 4.1 (and to a greater extend in the Appendix), we give an analysis of the effect of weights w_g and w_v on group behavior.

3.6. Merging

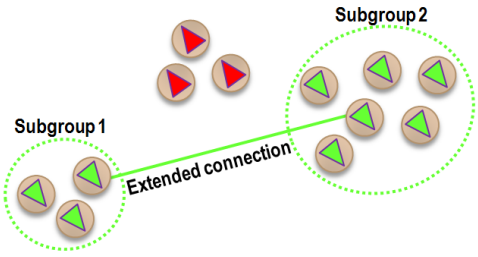


Figure 6: **Splitting/Merging** Agents of the green group split to avoid the agents of the red group. Setting $n^{ic} = 1$, an extended connection is made between agents of the two subgroups allowing them to merge back together directly after the end of the avoidance.

As introduced in Section 3.3 and demonstrated later in Section 4.1, group splitting can emerge. In some types of groups, such as families or small groups of friends, there is a high preference in members of groups to be attracted together. Though it is common to observe these types of groups split into subgroups for avoidance purposes, most of the time they merge back together as soon as they can; i.e., they aim in staying close to both nearby and far away group members. A similar constraint can be added to the model (Figure 6): in addition to selecting n^c connection neighbors, agents can select a limited number of "extended" connections n^{ic} from other subgroups (typically $n^{ic} \in [1,2]$). This allows agents to be attracted to both high influencing agents and to at least an agent from another subgroup and therefore guiding subgroups back together. By default, $n^{ic} = 0$ which means that subgroups will not seek to merge as soon as possible; rather they will be grouped after some time assuming they have the same goal. An example demonstrating this behavior can be seen in Section 4.2.

3.7. Preferred distance

In Equation 6 we have set three main constraints on speed to get the general group behaviors. The first constraint, $m^{des}(\mathbf{v})$ models the desire of an agent to move toward its goal. Adding the constraint $m^{VO}(\mathbf{v})$ results in extending the model with collision avoidance behavior. Similarly by adding $m^{VC}(\mathbf{v})$ grouping behavior is added. Following the same motif, by adding additional constraints an agent's behavior can become more complex. In this section we demonstrate such an application; we add a constraint on the minimum preferred distance agents want to keep from each other.

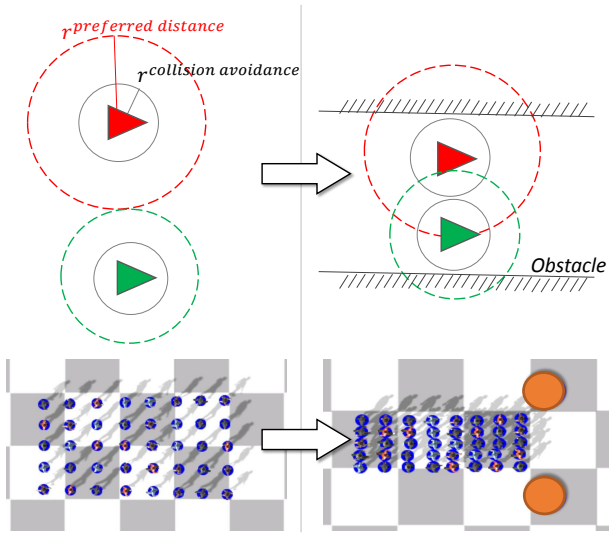


Figure 7: **Preferred distance** Agents have a new minimum distance constraint that is more flexible than collision avoidance, allowing them to keep some empty space around them whenever it is possible.

Keeping a minimum preferred distance is a behavior quite similar to collision avoidance. In both cases, a minimum distance with other agents is kept. The main difference is that in the case of collision avoidance, the minimum distance is strictly kept with others whereas the minimum preferred distance is a preference and can be ignored in specific situations, mainly when there is not enough space to satisfy it. In order to incorporate this behavior, we extend Equation 6 by adding a new constraint constraint $m^{VI}(\mathbf{v})$ using the

Table 2: **Tested Parameters** Parameters used in the sensitivity analysis; these affect the generated groups.

	Eqn.	Range	Value	Description
r^{max}	4,5	$[1-4]$	$1.2m$	Max. distance
w_g	6	$[0-.5]$	.5	Group weight
w_v	6	$[0-.5]$	.2	Des. velocity weight
n^c	8	$[1-45]$	3	Max. # of neighbors
$\|v_i\|$	12	$[.5-1.5]$	$1.33m/s$	Pref. Speed of agent A_i
e_{ij}	1	$[0-1]$	1	Influence of A_j to A_i

same formulation as $m^{VO}(\mathbf{v})$ (Equation 11) and increasing the radius. Additionally, to make this a more flexible constraint, we associate a very small weight $w_i \approx 0.1 * w_g$ to $m^{VI}(\mathbf{v})$. By doing this, agents will try to keep their distance from others but only if this is possible (see Figure 7). Therefore, Equation 6 can be replaced with the following:

$$m(\mathbf{v}) = m^{VO}(\mathbf{v}) + w_g m^{VC}(\mathbf{v}) + w_v m^{des}(\mathbf{v}) + w_i m^{VI}(\mathbf{v}). \quad (13)$$

4. Results

To evaluate the performance and quality of our system, we (a) test the effect of parameters on the results (Section 4.1) and (b) demonstrate example scenes with various types of groups (Section 4.2).

4.1. Parameter Analysis

The purpose of the sensitivity analysis is twofold; (a) to demonstrate that our framework is capable of generating a varying number of groups of different properties and (b) to demonstrate sets of parameters that can be used to generate desired group behaviors (such as the ones demonstrated in Section 4.2). To do so, we performed extensive tests on the effect of our system's parameters on different quantitative and qualitative characteristics of crowds for different scenarios. We identify four parameters that affect group behavior the most; the number of connection neighbors n^c, weights w_g and w_v for group and desired velocity preference respectively and finally the presence of leaders. We studied the effect of these parameters on various group properties such as cohesion, density, velocity, collisions and goal completion (see the Appendix for definitions).

4.1.1. Scenarios

The analysis is performed by simulating different scenarios that have the same basis, a group of agents where all agents have the same goal and equal weights in the relation matrix. During simulations, different kinds of internal or outside influences were introduced aiming to introduce perturbations to the group's behavior. Simulation were run multiple times (10 for each different case) for each different set of parameters with random initial positions for all agents. All of the simulations run long enough to capture the full effect of the perturbation. For all of the examined scenarios, all parameters except the ones being tested have default values (Table 2). We define three main scenarios for our experiments (Figure 8):

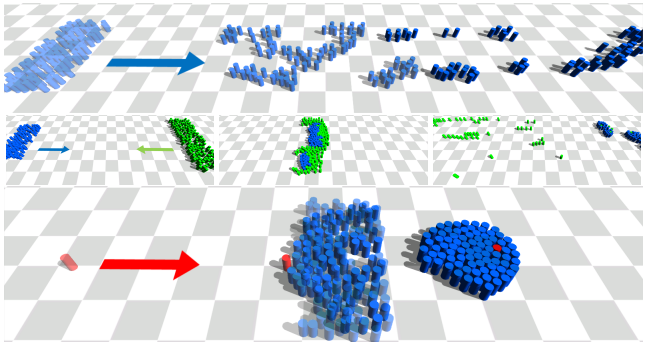


Figure 8: **Scenarios** The scenarios being used for the analysis. Blue indicates the agents being analyzed, green opposing agents and red leaders.(top) SimSpeed (center) SimCFlow (bottom) SimLeader.

1. **SimSpeed:** We initialize a group of 90 agents with different preferred velocities $\mathbf{v} : |\mathbf{v}| \sim U(.5, 1.5)m/s$.
2. **SimCFlow:** The outside influence is a set of 135 individual agents that move on the opposite flow of the group of 90 agents. Additionally, all agents have the same preferred speed $|\mathbf{v}| = 1.33m/s$.
3. **SimLeader:** We set the group size to 80 members, initially standing still without any goal (i.e., $|\mathbf{v}| = 0m/s$) and introduce a leader that aims in passing though them and attracting them to its own goal. To design a leader, we initialize the relation matrix such that the leader has a higher influence on other agents; in our experiments $e_{ij} = 1$ if A_j is a leader and $e_{ij} = .5$ otherwise.

For the first two scenarios we set the group to move from left to right whereas on the 3rd, agents start standing still.

4.1.2. Parameters Analysis

In this section we discuss the effect of some important parameters on observed group behaviors. A more thorough analysis can be found in the supplied Appendix.

Connection Neighbors The number of connection neighbors n^c has a big impact on group cohesion. As an example, consider how varying n^c affects group behavior for the *SimCFlow* scenario (Figure 9). Having a small n^c results in the splitting of the group into several small subgroups whereas progressively increasing n^c results in fewer and larger more cohesive groups. Setting $n^c = 45$ that corresponds to half the agents of the group, we get a very cohesive group where all group members stay together no matter the nature of the perturbation (different speeds or counter-flow of agents). Additionally, as expected and noticed in real life situations, having higher cohesion has a negative impact on speed or goal completion; agents need to slow down or temporary change directions so that they stay together. In the *SimSpeed* scenario for example, where agents are initialized with different desired speeds, having a small n^c ($[2:5]$) results in clusters of agents that move with different speeds whereas higher values ($n^c > 5$) result in larger, slower and more cohesive groups.

Weights Weights w_g and w_v (Equation 6) play an important role in emergent group behaviors; they aim in balancing between stay-

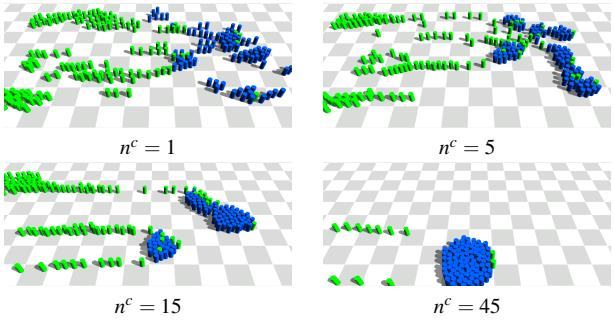


Figure 9: **Number of connections** Increasing the number of connections increases cohesion forcing clusters to stay together when interacting with a flow of agents (SimCFlow).

ing with group members and satisfying desired velocity respectively (Figure 10). Additionally, collision avoidance is affected indirectly; both w_g and w_v are always kept below 1 so that collision avoidance is always the most important criteria in selecting velocities. As expected, low values of w_g lead to poor group behavior whereas high values give good group behavior but also agents that move less efficiently towards their goals. Similarly, low values for w_v lead to agents that do not move efficiently towards their goals whereas high values forces them to prefer moving towards their targets in the most efficient way neglecting group behaviors. We found that a good balance between group and velocity preferences for most group behaviors is to set $w_g = 0.5$ and $w_v = 0.2$; here w_g is bigger than w_v and high enough to guide agents in staying together while still moving towards their goals.

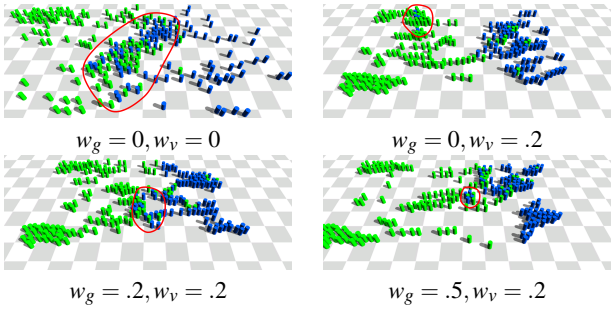


Figure 10: **Weights** When group weight w_g is large enough, the group does not break into clusters even in large interactions with other groups (SimCFlow).

Leaders/Followers In the leader-follower situation (*SimLeader*), some parameters have significant importance since they influence the leader's capacity to lead other agents. The most important parameter is the number of connections n^c. As we can see on the example in Figure 11, if n^c is close to zero, the leader agent will not be able to attract all of the agents since not all of them can perceive it. If on the other hand n^c is large, the leader will not have enough influence on the other agents to attract them; the group prefers to stay still. We found that an intermediate value of n^c around 4 optimizes the leader's capacity to lead other agents; this is the only situation in Figure 11 where the leader is able to lead all the other agents. On the other hand, weights w_g and w_v have similar influence to the other scenarios; followers have no goals which practically means w_v does not play any role in their behavior contrary to the leader. If the leader agent favors group behavior (w_g), then it will be attracted by the "no goal" heavy group and will fail to achieve its goal. If the leader favors velocity preference, then it will lose the majority of the group members whilst moving as efficiently as possible towards its goal.

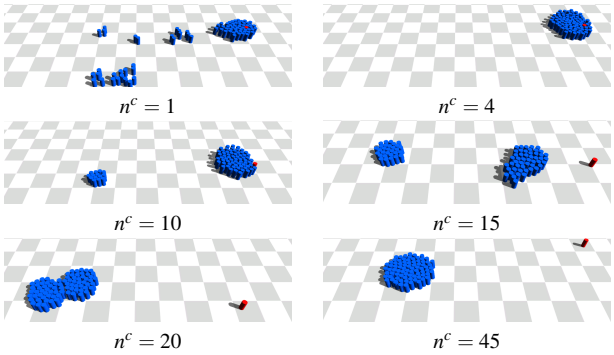


Figure 11: **Leader/Followers** Resulting groups for different n^c. Increasing the number of connections neighbors makes the standing group more rigid and difficult to move using a single leader.

4.2. Example Cases

Based on the results of the previous paragraphs, we designed some scenes consisting of multiple types of groups; these include tours with guides, small social groups, big cohesive groups, etc. Please consult Figure 1 for an overview of some of the group behaviors we discuss here and the supplemented video for group animations (and implementation details) in various environments. Here we discuss the parameters used to achieve these behaviors. For all cases, default values are the ones shown in Table 2 unless specified. In both example scenes, we can see some or all of the following types of groups at the same time.

Pedestrians We introduce groups of 2-4 agents walking together in streets as in typical real life environments. To achieve small cohesive groups we set $r^{max} = 1$ and $n^c = 1$ for groups of 2-3 agents and $n^c = 2$ for 4 agents (Figure 1b); i.e., each character only considers friends for group navigation. Additionally, we used the abreast line constrained (as described in Section 3.4) so that agents maintain side-by-side walking behavior even when interacting with other agents. In Figure 5 we show a comparison between using isotropic and anisotropic constraints.

Artist Performance Our system makes it easy to design a street performance (Figure 1a); people gather around a performer and stop to watch her performance. To achieve this, we set the performer as a leader but increase its r^{min} to 4 and set $r^{max} = 5$. This enforces a personal space around the performer where other agents in the group cannot enter. Additionally we set $n^c = 2$ for all agents so that agents in the same groups stay together.

Guided Tourists Guiding a group of tourists using a guide (Figure 1c) can be achieved by setting low n^c for the tourists so that they

stay with their closest friends whilst keeping the group intact. As discussed in Section 4.1 and the Appendix, setting $n^c \in [3-7]$ gives the optimal results; here we set $n^c = 3$. Additionally, the guide has influence $e_{ij} = 1$ to all other members of the group and the tourists have an influence of .5 to all other members including the guide. Finally, when constructing the VC with the respect to the guide, a guide constraint is used (Figure 4b) to keep the tourists staying behind the guide.

Students/Teachers Similarly to the guided tourist example, we set a higher influence of the teachers to the students (1 to .5 of the students) and set the relation matrix as seen in the video (Figure 1d). Students have no goal, i.e. their velocity will be zero if there are no teachers nearby, and are led by the teacher. By employing the queue constraint between students and a abreast constraint between student and teacher, students will walk in a queue formation with the teacher being on the side.

Crosswalk We also demonstrate two different types of groups crossing a crosswalk (Figure 12); two big rigid groups (left) with a queue constraint and two groups consisting of multiple small groups of 2-3 agents (right) using a circular constraint. In both cases, $n^c = 1$. In the first case, groups naturally form lanes to avoid each other more efficiently whereas on the second case the groups mix together but do not split from their groups.

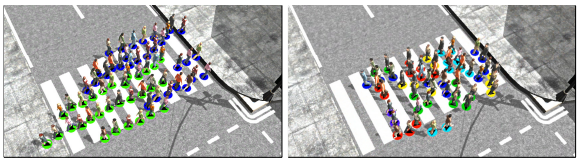


Figure 12: **Demo: Crossing**

Switching between groups We are also able to simulate followers that switch between different leaders. These kind of situations can be observed in tourist attractions where tourists might switch to groups that they consider more interesting (interest is the influence of each leader). Another interesting application could be a hero unit in a game which when it acquires more experience/influence, can attract units from another hero unit. This phenomenon can naturally emerge with our approach by setting a relation matrix where multiple leaders have non zero influences on the same follower. We show an example of this behavior in Figure 13; the yellow agents following leader A at the beginning switch to following leader B, since B has higher influence to them than A does.

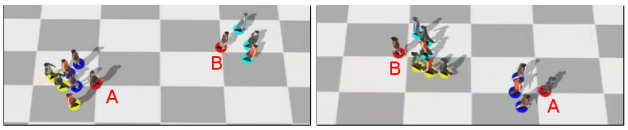


Figure 13: **Demo: Switching Groups**

Dynamically Switching Parameters As described in Section 3, a user defines the relation matrix and relevant parameters at a preprocessing step. This does not imply that parameters cannot change at runtime; parameters can be switched during simulation time to change group behavior. We demonstrate such a scenario here (Figure 14) where a group in a square formation changes at runtime to a line formation to pass through a corridor more efficiently and then changes back to the original parameters (and therefore the square formation). The square formation is setup using $n^c = 3$ and $r^{max} = 1$; the front agents use the abreast constraint whereas the others use the line constraint (the relation matrix can be seen in the supplied video). To switch to the queue formation, we set $r^{max} = 2$ and set the queue constraint to the front agents.

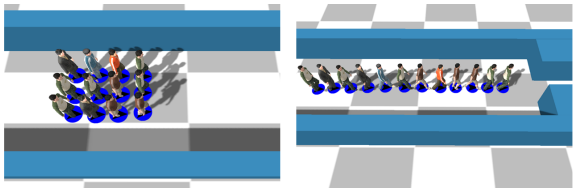


Figure 14: **Demo: Dynamically Switching Parameters**

Inter-group interaction During group interactions, groups can bypass each other or one can split spatially into subgroups. Assuming a small number of connection neighbors n^c, these subgroups might not merge back again soon. If desired, group re-merging can be enforced as presented in Section 3.6. In Figure 15 we demonstrate that our approach is able to reproduce these different behaviors with the right set of parameters. Setting $n^c = 3$ to the group on the left, results to a big cohesive group that bypasses the small one. Decreasing n^c to 2, makes the group split to avoid the small one. If the large group does not consider agents from both subgroups, it will remain split for a long time. To encourage re-merging, we force each agent to consider at least another agent from the other subgroup by setting $n^{ic} = 1$. For all cases, the small group had $n^c = 3$.

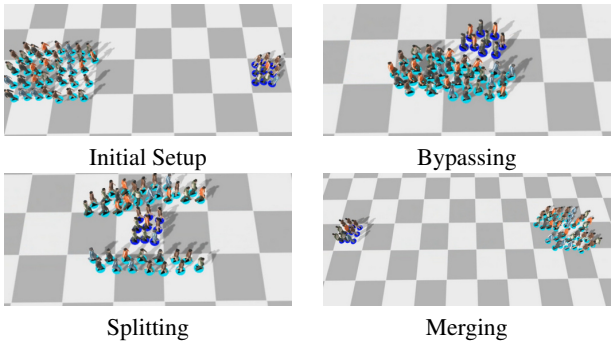


Figure 15: **Group Interactions** Different interaction behaviors can emerge depending on parameters.

5. Discussion and Limitations

We propose a crowd simulation algorithm suitable for both individuals and groups by extending a velocity-based approach with new kinds of interactions between agents. These local interactions enable two or more agents to move while trying to stay below a bounded distance from each other. Agents select other agents to stay close to based on a relation matrix that is easily set up by a designer. This bottom-up way to generate group behaviors gives us

a very flexible solution introducing significant progress beyond the state of the art in group simulation.

It is a unified solution to group simulation, since we can simulate a large number of different group situations just by setting appropriate parameters. This approach can simulate such diverse groups as couples of walkers, lanes, formations and tourists with complex structure (e.g., group guide with different families in the group, etc). We demonstrate crowds made of mixtures of groups of different types as well as individual agents; all using the same algorithm. We show some intuitive correlations between the parameters space and the visual aspect of moving groups; this allows a designer to setup a simulation for a given desired scenario by selecting appropriate parameters. Finally, the flexibility of the solution produces interesting patterns and emergent behaviors. From the simple behaviors of staying together and avoiding collision, simulated agents demonstrate the ability to reproduce real groups behaviors such as group splitting, flow smoothing, group switching, fast agents slowing down to wait for slower group members, etc.

Our approach however has some limitations. First, our definition of local interactions in a group is a unique one: stay within a distance (or polygonal area). We would like to consider some other types of interactions such as holding hand that add a very strong navigational constraint or more complex positioning with preferred position instead of hard constrained one. Moreover, we would like to add some higher level behaviors such as members of a group that meet and gather somewhere. As we show with the formation constraint (Section 3.4) and the merging (Section 3.6), constraints can easily be added to the velocity computation or neighbors selection to handle more complex behaviors.

We would also like to evaluate the level of realism that can be achieved by the proposed system. This is a very difficult task since reference and observations data on groups are limited. The design of relevant evaluation metrics for groups moving in crowds needs addressing as well. This by itself is an entire problem, certainly out of the scope of this paper, and which requires a large effort in performing relevant evaluation. Finally, given the proportion of groups in real crowds ($> 70\%$ move in groups in typical crowds [MPG*10]), it is certainly a promising research path for considering other type of applications, like architecture design and safety.

6. Conclusions and Future Work

We have presented a new microscopic algorithm to simulate diverse types of groups in crowds using a set of intuitive parameters. We have three main contributions. Firstly, we formulate a new model of local interactions between agents so that they form groups of agents moving together. We have demonstrated the ability of this local model to handle different kind of groups (couples to very large groups, leader(s)/followers groups, formation walking groups, ...) as well as a wide variety of behaviors (various levels of cohesion, group splitting and merging, group switching, ...). Secondly, we proposed a general, unified, crowd simulation algorithm which considers mixtures of groups and individual agents moving in the same scene. A simple relation matrix and few parameters determine the whole crowd in an intuitive way. Finally, we present an empirical mapping between the parameter space of our solution and emergent group behaviors: this makes the parameter tuning stage easy with respect to the desired simulation scenario. Our results section demonstrates that complex scenarios can be achieved with relative ease.

Our results open promising future research paths. Our priority is to extend our solution with other new types of interactions between agents of the same group and new types of groups. In reality, groups are places of possibly complex relations and interaction between members and also group compositions can evolve in time. We demonstrated that we can switch parameters in simulation time, resulting in different group behaviors; by introducing morphing abilities we could have more control on group transitions. Additionally, by introducing complex interactions we would again extend the validity domain of our solution. Finally, an interesting avenue would be to introduce models of local interactions between 2 groups (or more) or between groups and individuals. Many interesting situations involve these kinds of interaction, like groups of teammates from opposing teams, or specific behaviors of individuals with groups [BOP15].

Finally, our more long term objective is to evaluate our model from real crowd data. Some recent tracking techniques [PEVG10] or motion analysis techniques [CKGC14] are able to detect groups in crowd motions. These techniques could be put into practice to create a database on group behaviors and their influence on traffic. This would certainly serve as a basis to address the problem of evaluating the level of realism of our solution.

Acknowledgements

Work for this work has been funded by ANR JCJC PERCOLATION project, fund number ANR-13-JS02-0008.

References

[AMBR*12] ALONSO-MORA J., BREITENMOSER A., RUFLI M., SIEGWART R., BEARDSLEY P.: Image and animation display with multiple mobile robots. *The International Journal of Robotics Research 31*, 6 (2012), 753–773. 2

[BMdOB03] BRAUN A., MUSSE S. R., DE OLIVEIRA L. P. L., BODMANN B. E.: Modeling individual behaviors in crowd simulation. In *Computer Animation and Social Agents, 2003. 16th International Conference on* (2003), IEEE, pp. 143–148. 2

[BOP15] BRUNEAU J., OLIVIER A.-H., PETTRÉ J.: Going through, going around: A study on individual avoidance of groups. *Visualization and Computer Graphics, IEEE Transactions on 21*, 4 (2015), 520–528. 2, 3, 10

[CC14] CHARALAMBOUS P., CHRYSANTHOU Y.: The PAG Crowd: A Graph Based Approach for Efficient Data-Driven Crowd Simulation. *Computer Graphics Forum 33*, 8 (2014), 95–108. 2

[CKGC14] CHARALAMBOUS P., KARAMOUZAS I., GUY S. J., CHRYSANTHOU Y.: A Data-Driven Framework for Visual Crowd Analysis. *Computer Graphics Forum 33*, 7 (2014), 41–50. 10

[FS98] FIORINI P., SHILLER Z.: Motion planning in dynamic environments using velocity obstacles. *The International Journal of Robotics Research 17*, 7 (1998), 760–772. 2

[GCC*10] GUY S. J., CHHUGANI J., CURTIS S., DUBEY P., LIN M., MANOCHA D.: PLEdestrians: a least-effort approach to crowd simulation. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics*

Symposium on Computer Animation (2010), Eurographics Association, pp. 119–128. 2

[GKLM11] GUY S. J., KIM S., LIN M. C., MANOCHA D.: Simulating heterogeneous crowd behaviors using personality trait theory. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2011), ACM, pp. 43–52. 2

[HM95] HELBING D., MOLNAR P.: Social force model for pedestrian dynamics. *Physical review E 51*, 5 (1995), 4282. 2

[JCP*10] JU E., CHOI M. G., PARK M., LEE J., LEE K. H., TAKAHASHI S.: Morphable crowds. In *ACM Transactions on Graphics (TOG)* (2010), vol. 29, ACM, p. 140. 2

[KDB12] KIMMEL A., DOBSON A., BEKRIS K.: Maintaining team coherence under the velocity obstacle framework. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1* (2012), International Foundation for Autonomous Agents and Multiagent Systems, pp. 247–256. 2, 3, 5

[KGL*13] KIM S., GUY S. J., LIU W., LAU R. W., LIN M. C., MANOCHA D.: Predicting pedestrian trajectories using velocity-space reasoning. In *Algorithmic Foundations of Robotics X*. Springer, 2013, pp. 609–623. 2

[KLLT08] KWON T., LEE K. H., LEE J., TAKAHASHI S.: Group motion editing. In *ACM Transactions on Graphics (TOG)* (2008), vol. 27, ACM, p. 80. 2

[KO12] KARAMOUZAS I., OVERMARS M.: Simulating and evaluating the local behavior of small pedestrian groups. *Visualization and Computer Graphics, IEEE Transactions on 18*, 3 (2012), 394–406. 2

[LA15] LEMERCIER S., AUBERLET J.-M.: Towards more behaviours in crowd simulation. *Computer Animation and Virtual Worlds* (2015). 2, 3

[LCHL07] LEE K. H., CHOI M. G., HONG Q., LEE J.: Group behavior from video: A data-driven approach to crowd simulation. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2007), SCA '07, Eurographics Association, pp. 109–118. 2

[LCL07] LERNER A., CHRYSANTHOU Y., LISCHINSKI D.: Crowds by example. In *Computer Graphics Forum* (2007), vol. 26, Wiley Online Library, pp. 655–664. 2

[LMM03] LOSCOS C., MARCHAL D., MEYER A.: Intuitive crowd behavior in dense urban environments using local laws. In *Theory and Practice of Computer Graphics, 2003. Proceedings* (2003), IEEE, pp. 122–129. 2

[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. 1, 2, 10

[MT97] MUSSE S. R., THALMANN D.: *A model of human crowd behavior: Group inter-relationship and collision detection analysis*. Springer, 1997. 2

[MT01] MUSSE S. R., THALMANN D.: Hierarchical model for real time simulation of virtual human crowds. *Visualization and Computer Graphics, IEEE Transactions on 7*, 2 (2001), 152–164. 2

[NGCL09] NARAIN R., GOLAS A., CURTIS S., LIN M. C.: Aggregate dynamics for dense crowd simulation. In *ACM Transactions on Graphics (TOG)* (2009), vol. 28, ACM, p. 122. 2

[OPOD10] ONDŘEJ J., PETTRÉ J., OLIVIER A.-H., DONIKIAN S.: A synthetic-vision based steering approach for crowd simulation. In *ACM Transactions on Graphics (TOG)* (2010), vol. 29, ACM, p. 123. 2

[PE09] PETERS C., ENNIS C.: Modeling groups of plausible virtual pedestrians. *IEEE Computer Graphics and Applications*, 4 (2009), 54–63. 2

[PEVG10] PELLEGRINI S., ESS A., VAN GOOL L.: Improving data association by joint modeling of pedestrian trajectories and groupings. In *Computer Vision–ECCV 2010*. Springer, 2010, pp. 452–465. 10

[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 *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2009), ACM, pp. 189–198. 2

[PV08] PEDICA C., VILHJÁLMSSON H.: Social perception and steering for online avatars. In *Intelligent Virtual Agents* (2008), Springer, pp. 104–116. 2

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

[Rey87] REYNOLDS C. W.: Flocks, herds and schools: A distributed behavioral model. *ACM Siggraph Computer Graphics 21*, 4 (1987), 25–34. 2

[Rey99] REYNOLDS C. W.: Steering behaviors for autonomous characters. In *Game developers conference* (1999), vol. 1999, pp. 763–782. 2

[RY13] ROJAS F. A., YANG H. S.: Immersive human-in-the-loop hmd evaluation of dynamic group behavior in a pedestrian crowd simulation that uses group agent-based steering. In *Proceedings of the 12th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry* (2013), ACM, pp. 31–40. 2

[SSKF10] SCHUERMAN M., SINGH S., KAPADIA M., FALOUTSOS P.: Situation agents: agent-based externalized steering logic. *Computer Animation and Virtual Worlds 21*, 3-4 (2010), 267–276. 2, 3

[TCP06] TREUILLE A., COOPER S., POPOVIĆ Z.: Continuum crowds. *ACM Transactions on Graphics (TOG) 25*, 3 (2006), 1160–1168. 2

[TYK*09] TAKAHASHI S., YOSHIDA K., KWON T., LEE K. H., LEE J., SHIN S. Y.: Spectral-based group formation control. In *Computer Graphics Forum* (2009), vol. 28, Wiley Online Library, pp. 639–648. 2

[VDBGLM11] VAN DEN BERG J., GUY S. J., LIN M., MANOCHA D.: Reciprocal n-body collision avoidance. In *Robotics research*. Springer, 2011, pp. 3–19. 2

[VdBLM08] VAN DEN BERG J., LIN M., MANOCHA D.: Reciprocal velocity obstacles for real-time multi-agent navigation. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on* (2008), IEEE, pp. 1928–1935. 2, 3, 4

[VMdO03] VILLAMIL M. B., MUSSE S. R., DE OLIVEIRA L. P. L.: A model for generating and animating groups of virtual agents. In *Intelligent Virtual Agents* (2003), Springer, pp. 164–169. 2

[YCP*08] YEH H., CURTIS S., PATIL S., VAN DEN BERG J., MANOCHA D., LIN M.: Composite agents. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2008), Eurographics Association, pp. 39–47. 2