Proxemic Group Behaviors using Reciprocal Multi-Agent Navigation

Liang He[1] and Jia Pan[2] and Wenping Wang[2] and Dinesh Manocha[1]
http://gamma.cs.unc.edu/Proxemic

***Abstract*— We present a decentralized algorithm for group-based coherent and reciprocal multi-agent navigation. In addition to generating collision-free trajectories for each agent, our approach is able to simulate macroscopic group movements and proxemic behaviors that result in coherent navigation. Our approach is general, makes no assumptions about the size or shape of the group, and can generate smooth trajectories for the agents. Furthermore, it can dynamically adapt to obstacles or the behavior of other agents. The additional overhead of generating proxemic group behaviors is relatively small and our approach can simulate hundreds of agents in real-time. We highlight its benefits on different benchmarks.**

I. Introduction

Multi-agent navigation algorithms are widely used for motion planning among static and dynamic obstacles [1], for autonomous localization [2], and in the simulation of animated characters or human crowds in games and virtual worlds [3]. A key issue in these applications is collision-free path planning: given a set of agents in a static or a dynamic environment, each with its own initial and goal positions, computing collision-free trajectories for each agent towards its goal. In some scenarios, there are additional constraints imposed on the trajectories corresponding to dynamics, biomechanics, or human-like behaviors.

There is extensive literature on multi-agent navigation and path planning. Most prior approaches treat each agent as an independent entity or unit in terms of computing its trajectory. However, in some challenging scenarios or cluttered environments, these single-agent based navigation algorithms are unable to compute collision-free trajectories, result in unnatural behaviors, or result in trajectories that are not smooth. These problems are more noticeable in dense environments with a high number of agents or obstacles, which can significantly restrict each agent's movement [4], [3].

In many applications we observe group-based behaviors, in which many nearby agents exhibit similar movements or trajectories. Such behaviors are frequently observed in human crowds [5], artificial life [6], swarm robotics, etc. There is considerable work on simulating group-based multi-agent behaviors. Many real-time algorithms are based on decentralized methods and use a combination of local and global navigation methods to simulate a large number of agents [7], [8], [9]. These algorithms have been used to simulate coherent movements of agents while avoiding static obstacles. However, some of these methods do not take into account reactive behaviors [10] between different groups. Other techniques assume that the group units or their sizes are fixed, or are unable to generate smooth trajectories. In this paper, we address the problem of group-based coherent and reciprocal multi-agent navigation. A group corresponds to two or more agents that interact to achieve a shared goal. This includes simulating the group movements and their *proxemic* behaviors, i.e human spatial behavior during social interactions [5]. Proxemic behaviors have been extensively studied in social psychology by observing human crowds [11]. One key observation is that in high-density conditions, the spatial distribution of a group is characterized by the presence of a leader who guides the other members in crossing the space in a river-like pattern [12]. As a result, nearby agents tend to move as a coherent group instead of as independent individuals. However, current multi-agent navigation algorithms are unable to generate such trajectory behaviors.

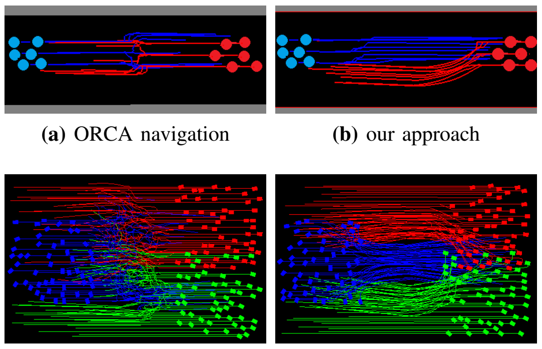
(a) ORCA navigation (b) our approach (c) ORCA navigation (d) our approach

Fig. 1: The comparison between the trajectories generated by ORCA algorithm [10] and our approach, for both holonomic agents (top row) and non-holonomic agents (bottom row). ORCA is a single-agent based navigation algorithm that can't generate proxemic group behaviors. In contrast, our approach can generate smooth agent trajectories and coherent navigation. We use different colors to indicate each agent' group association, which is not observable to other agents.

Main Results: We present a novel algorithm for simulating proxemic group behaviors using decentralized multi-agent navigation. Our formulation is general, makes no assumptions about the groups, and can generate smooth macroscopic proxemic behaviors for a group of agents.

Our real-time algorithm computes the groups for each agent during each cycle of the simulation. It classifies the

[1] Liang He and Dinesh Manocha are with the Department of Computer Science, the University of North Carolina at Chapel Hill; [2] Jia Pan and Wenping Wang are with the Department of Computer Science, the University of Hong Kong

This research is supported in part by ARO Contract W911NF-14-1-0437 and NSF award 1305286.

neighboring agents into two types of groups: a *proxemic group* that the agent tries to follow, and *obstacle groups* that are bypassed by that agent. We use a velocity-space formulation and reduce the velocity computation problem for each agent to a multi-objective constrained optimization problem. This includes the velocity-obstacle based constraints for collision-free navigation and additional connectivity constraints to generate coherent movements. We use a consistent bypassing scheme so that each agent avoids collisions with the obstacle groups. We also use a dynamic following strategy among the agents in the proxemic group. We demonstrate the performance of our algorithm on multiple scenarios and also compare the performance with prior multi-agent reciprocal collision avoidance algorithms.

As compared to prior multi-agent navigation algorithms, our approach offers the following benefits:

- Our formulation is general and can handle arbitrarily sized or shaped groups.
- The proxemic group behaviors are dynamically generated and our approach can adapt to the obstacles and to the behaviors of other agents.
- Our algorithm can generate smooth trajectories and perform coherent navigation for different groups (as shown in Figure 1).
- The additional runtime overhead over a single-agent based reciprocal collision avoidance algorithm is at most 14%. Our approach can generate collision-free trajectories as well as macro-level proxemic behaviors.

The rest of the paper is organized in the following manner. We give a brief overview of prior multi-agent navigation algorithms in Section II. We introduce our notation and give an overview of our approach in Section III. We describe the group-based multi-agent navigation algorithm in Section IV and highlight its performance in Section V.

II. Related Work

There is extensive work on distributed multi-agent navigation algorithms. In this section, we mainly classify them based on single agent-based and group-based navigation strategies.

Most approaches decompose the multi-agent motion planning problem into a set of independent agent planning subproblems. This greatly reduces the complexity of each subproblem and enables the use of single-agent global path planners [1] to compute the preferred velocity or the preferred direction of motion for each agent. One main issue in such distributed planning methods is how to locally adapt each agent's movement to avoid collisions with nearby agents and dynamic obstacles. Some of the commonly used techniques are based on potential fields [13], dynamic windows [14], and reciprocal velocity obstacles (RVO) [10] and its variants [15], [16]. However, such pairwise computation methods may fail to find a feasible solution for all agents in cluttered situations because the agents may block each other along their respective directions of motion. Some recent techniques handle such scenarios by reducing their velocity [17], [18] in dense settings.

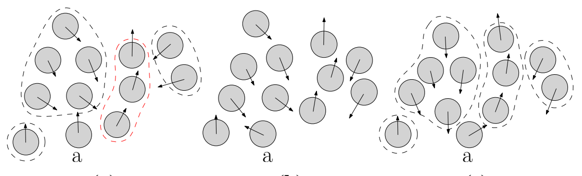

Fig. 2: Comparison between reciprocal collision avoidance without and with proxemic behavior. (a) is the crowd at time t_0, and (b), (c) are the crowd at time t_1 for navigation without and with proxemic behaviors, respectively. In the crowd with proxemic behavior, the groups in the previous time t_0 will be approximately maintained, while in the crowd without proxemic behavior, each agent will move independently.

Some recent methods solved the distributed multi-agent planning problem by clustering different agents into groups. Curtis et al. [19] and Krontiris et al. [20] re-arrange the shapes of these groups for better utilization of the free space. Kimmel et al. [21] control the positions of group members so that they can walk around static obstacles. Santos et al. [22] extend RVO to enable reciprocal avoidance among the groups by considering each group as a super-agent. This method requires a pre-defined group assignment, and the group index of each agent must be observable to its nearby agents. He et al. [7] present a strategy to enable an agent to walk around the aggregated agents, but it is not able to generate proxemic navigation behaviors. There is considerable work on continuum crowd simulation algorithms that use continuous mathematical models based on fluid or granular flows for medium and high-density crowds. These models have also been combined with discrete agent methods [3], [23] to simulate crowds of varying densities. Our approach is orthogonal to these methods, uses only discrete level agent representations and can generate smooth trajectories and smooth navigation flows for nearby agents.

III. Overview

In this section, we first introduce our notation and give an overview of our approach.

A. Problem Definition

Our goal is to simulate macro-level proxemic behaviors for a group of agents using decentralized multi-agent, micro-level, navigation algorithms. This problem can be formally defined as follows. We take as given a set of n decision-making agents sharing a (2D) environment consisting of obstacles. For simplicity, we assume the geometric shape of each agent a is represented as a disc of radius r_a, and its current position $\mathbf{p}_a$ and velocity $\mathbf{v}_a$ are observable to nearby agents. Each agent employs a continual cycle of sensing and acting with a time period τ. During each cycle, the agent observes nearby agents and obstacles within a certain neighborhood, and tends to compute a local trajectory towards its goal position $\mathbf{g}_a$. The trajectory should be collision-free and also satisfy other constraints. Each agent is assumed to have a preferred velocity $\mathbf{v}_a^{\text{pref}}$, and this is the velocity at which the agent would travel if there were no other agents or obstacles in the environment. Prior multi-agent navigation algorithms based on velocity obstacles [24], [10] compute a new velocity $\mathbf{v}_a^{\text{new}}$ using constrained optimization. The velocity-level constraints are used to compute the new collision-free velocity and we augment them with additional constraints to generate macro-scale proxemic behaviors.

1) Micro-level Velocity Constraints: The micro-level or ORCA constraints are used to specify the space of velocities that can guarantee a collision-free motion during τ [10]. The constraints are represented as the boundary of a half plane containing the space of feasible, collision-free velocities. Given two agents a and b, we compute the minimum vector $\mathbf{u}$ of the change in relative velocity needed to avoid collisions. This constraint can be guaranteed by requiring each agent to change its current velocity by at least $1/2\mathbf{u}$, and can be expressed as:

$$ORCA_{a|b} = \{\mathbf{v} | (\mathbf{v} - (\mathbf{v}_a + \frac{1}{2}\mathbf{u})) \cdot \hat{\mathbf{u}} \geq 0\}, \qquad (1)$$

where $\hat{\mathbf{u}}$ is the normalized vector of $\mathbf{u}$. All of the neighboring agent a's neighbors will impose similar ORCA constraints on a's velocity.

2) Macro-level Proxemic Behaviors: The proxemic behavior imposes additional restrictions on the agents' movement, and we formalize these restrictions as macro-level driving functions (MDF) defined over the pedestrians. These MDFs are used to compute the velocities that are able to achieve the coherent movement of clustered agents, which have similar positions and velocities. Such proxemic behavior can be formally described by leveraging the concept of *connected agents*. Given two agents a and b, we say they are connected if both of them belong to the same group G, and they can move towards each other along a straight line connecting them without colliding with any other agents outside the group G, i.e. $\forall c \notin G, \overline{\mathbf{p}_a\mathbf{p}_b} \cap (c \oplus \mathcal{D}(\mathbf{0}, r_a)) = \emptyset$, where $\mathcal{D}(\mathbf{0}, r_a)$ is a disc centered at the origin with radius r_a, and $\overline{\mathbf{p}_a\mathbf{p}_b}$ is the line segment connecting two agents' center $\mathbf{p}_a$ and $\mathbf{p}_b$. We denote the connected relationship between a and b as a function

$$\text{conn}(a,b) = \begin{cases} 1 & \text{if } a \text{ and } b \text{ are connected} \\ 0 & \text{otherwise.} \end{cases} \qquad (2)$$

A group G is connected if and only if each of its members has at least one fellow agent (in G) to which it can be connected:

$$\text{conn}(G) = \begin{cases} 1 & \text{if } \forall a \in G, \exists b \in G, \text{conn}(a,b) = 1 \\ 0 & \text{otherwise.} \end{cases} \qquad (3)$$

In order to generate proxemic behaviors and minimize the collisions between the agents, each agent should geometrically stay inside a group that is well connected. In this way, all group members can easily make progress without too much effort by following a leader and walking along the group's flow. This guarantees that every group member has at least one feasible moving strategy in a dense environment. Based on the concept of connected agents, the corresponding multi-agent navigation problem can be formalized as follows: given the groups $G^1, ..., G^n$ defined over the agents, how to perform collision-free navigation and keep each of them connected whenever possible during the navigation, i.e. how to maximize the MDF $\sum_{G^i} \text{conn}(G^i)$. Note that unlike the micro-level constraints, this macro-level formulation is defined over the entire set of agents. We present a decentralized algorithm to compute a solution to this problem.

3) Velocity Computation Problem: We can summarize our new velocity computation problem for each agent as follows: Given agents $\{a\}$ and their velocities $\{\mathbf{v}_a\}$, we can formulate our multi-agent navigation as a multi-objective constrained optimization problem. Formally,

$$\{\mathbf{v}_a^{\text{new}}\} = \begin{cases} \arg\min \|\mathbf{v}_a - \mathbf{v}_a^{\text{pref}}\|, \forall a \\ \arg\max \sum_{G^i} \text{conn}(G^i), \end{cases} \qquad (4)$$

subject to the ORCA constraints: $\forall a, \mathbf{v}_a \in \bigcap_{b \neq a} ORCA_{a|b}$.

B. Our Approach

In order to generate the macro-level proxemic group behaviors in a decentralized manner, each agent a initially clusters its neighboring agents into a set of groups $\{G_a^j\}$. This clustering result is used to infer how its neighbors are grouped with respect to the rest of the agent groups, $\{G^i\}$, and then used to adapt the velocity of a to stay within $\{G_a^j\}$. In particular, the agent a first observes its neighborhood in terms of the positions and velocities of all agents within a certain radius, and then clusters the neighboring agents with similar positions and velocities into different groups $\{G_a^j\}$. Each of these groups, $\{G_a^j\}$, has a close proximity to a. This clustering only requires the local information about other agents around a and can be computed in a decentralized manner. We associate a group velocity $\mathbf{v}_{G_a^j}$ with each group, and that group velocity is equal to the average current velocity of all the agents in that group. The agent a will select to join one of the groups whose direction of motion is most similar to a's current preferred velocity. This group is called the *proxemic group* and is denoted as G_a^*. Its elements are called the *proxemic agents* with respect to a. All other groups are considered as obstacles with a time-variable shape moving with the corresponding group velocity. These groups are called the *obstacle groups* and their members are *obstacle agents*. Next, the agent a computes the local trajectory that can generate proxemic behavior in two steps. The first step concerns the *inter-group proxemic avoidance* where a makes a high level decision about whether to bypass each of the obstacle groups from the left side or from the right side. The side on which to bypass is selected in such a way that the entire proxemic group G_a^* will likely bypass the obstacle groups in a consistent manner without splitting or mixing with other groups. The second step relates to the *intra-group proxemic behavior* in which the agent a selects one proxemic agent to follow in order to make progress towards its goal and agent a maintains the proxemic group behavior. After that, the agent can compute its preferred velocity and use that to obtain an actual velocity that satisfies the micro-level ORCA constraints. Figure 4b illustrates the difference between navigation with and without proxemic behaviors.

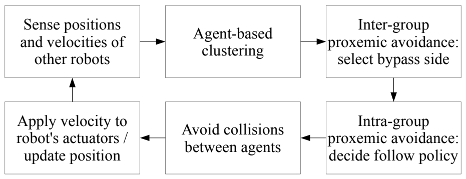
Sense positions and velocities of other robots
Agent-based clustering
Inter-group proxemic avoidance: select bypass side
Intra-group proxemic avoidance: decide follow policy
Avoid collisions between agents
Apply velocity to robot's actuators / update position

Fig. 3: The pipeline of our algorithm.

IV. Group-based Multi-Agent Navigation

In this section, we present our algorithm for group-based multi-agent navigation. The goal is to compute coherent movement for all agents within a group so as to simulate the proxemic behaviors. Fundamentally, we compute a solution to the multi-objective optimization problem stated in Equation 4. We describe our decentralized approach for a single agent a, whose preferred velocity is given as input, and we compute the actual velocity using the optimization algorithm. Our solution consists of multiple steps. First, we cluster a's neighborhood into a proxemic group G^*_a and several obstacle groups. The agent then chooses a high-level navigation policy (left or right) to bypass the obstacle groups in such a way that if all proxemic agents consistently use the same policy, then the entire proxemic group would bypass obstacle groups as a whole. Next, the agent a decides how to follow one proxemic agent within the proxemic group G^*_a to make progress toward the goal. Finally, the actual velocity is computed that satisfies all these constraints. The pipeline for our approach is shown in Figure 3.

A. Agent-based Clustering

Let N_a be the set of neighboring agents that are currently within the local neighborhood of the agent a: $N_a = \{b \mid \|\mathbf{p}_b - \mathbf{p}_a\| < n_a\}$, where n_a is the radius of a's neighborhood. Then we cluster the neighboring agents into multiple groups, $\{G^j_a\}$, according to their positions and velocities.

As computed, the agent groups $\{G^j_a\}$ can be classified into two types. Some of them are of the similar direction of motion as the agent a, and some of them may collide with the agent a in the near future. We assume the shape of each group is given by the convex hull $\mathcal{CH}(G)$ of the set of agents constituting the group G, and that the group's velocity is the average velocity of all member agents and represented as $\mathbf{v}_G$. Given this information, the agent a will choose to join the group whose velocity is closest to the agent a's current velocity: $G^*_a = \arg\max_{G^i_a, \mathbf{v}_a \cdot \mathbf{v}_{G^i_a} \geq 0} \mathbf{v}_a \cdot \mathbf{v}_{G^i_a}$. G^*_a is the proxemic group, and all other groups are the obstacle groups.

B. Inter-Group Proxemic Avoidance: Bypass Side Selection

Next, the agent determines the suitable side on which to bypass the obstacle groups, and we build on the concept of velocity obstacles [10] as shown in Figure 4a. Given a group G and its velocity $\mathbf{v}_G$, the velocity obstacle $VO_{a|G}$ for agent a induced by group G is defined as the set of agent a's velocities $\mathbf{v}_a$ that will result in a collision with G at some point within time window τ assuming that group G keeps

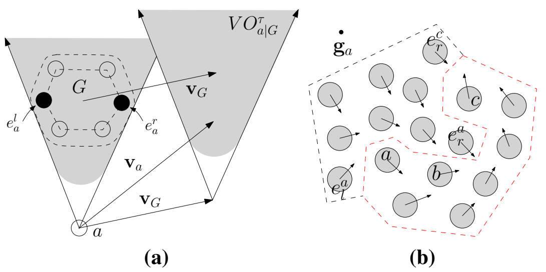

Fig. 4: (a) The velocity obstacle $VO^\tau_{a|G}$ for agent a induced by a group G of agents. If G only contains a single agent b, $VO_{a|G}$ reduces to the traditional velocity obstacle $VO^\tau_{a|b}$ induced by the agent b. The black agents e^l_a and e^r_a are the two most extreme agents in the group G. (b) Collision avoidance policy: The agent a's nearby agents are divided into two groups, encircled by the red and black dashed lines, respectively. To reach the goal position $\mathbf{g}_a$, a needs to bypass the obstacle group. For agent a, the two most extreme agents in the obstacle group are e^a_l and e^a_r. In this example, c will the leader and a, b will choose to be followers.

its velocity $\mathbf{v}_G$:

$$VO^\tau_{a|G} = \{\mathbf{v} | \exists t \in [0, \tau] \text{ such that } \mathbf{p}_a + (\mathbf{v} - \mathbf{v}_G)t \in \mathcal{CH}(G) \oplus \mathcal{D}(\mathbf{0}, r_a)\}, \quad (5)$$

where $\mathcal{D}(\mathbf{0}, r_a)$ is a disc centered at the origin with radius r_a. This equation implies that if agent a chooses a velocity outside the velocity obstacle $VO_{a|G}$, it will not collide with group G within the time window τ. As shown in Figure 4a, the convex hull $\mathcal{CH}(G)$ need not be computed explicitly, instead the velocity obstacle can be fully defined by the extreme agents in radial directions in the group, as observed from $\mathbf{p}_a$. We denote the most "clockwise" agent as e^r_a and the most "counterclockwise" agent as e^l_a. These two agents e^r_a and e^l_a are most important in terms of computing the avoidance trajectory of agent a.

For agents in the same group, they may choose different extreme agents from the same obstacle group, because they have different positions and velocities relative to the obstacle group. However, this will not be an issue as long as they select the same side (all e_l or e_r) while bypassing the obstacle group. If so, the entire group will keep moving coherently and will not collide with the other groups. To achieve this, the agent will use the group velocity to decide on which side of the obstacle group it needs to bypass. In particular, given the proxemic group G^*_a and one obstacle group G, let their average velocity vector be $\mathbf{v}_{G^*_a}$ and $\mathbf{v}_G$ and average positions be $\mathbf{p}_{G^*_a}$ and $\mathbf{p}_G$, respectively. The relative position and velocity between these two groups can be represented as $\mathbf{p}' = \mathbf{p}_{G^*_a} - \mathbf{p}_G$ and $\mathbf{v}' = \mathbf{v}_{G^*_a} - \mathbf{v}_G$ respectively. The side s the agent a should choose to avoid G can be computed according to the relationship between the normal $\mathbf{n}$ of the 2D plane and the cross product between $\mathbf{v}'$ and $\mathbf{p}'$

$$s = \begin{cases} \text{r (right)} & \text{if } (\mathbf{v}' \times \mathbf{p}') \cdot \mathbf{n} < 0 \\ \text{l (left)} & \text{otherwise.} \end{cases} \quad (6)$$

The solution of Equation 6 provides a rough direction of motion for each agent. Since the group shape can be nonconvex, and the convex hull of different groups may overlap with each other, we need a more sophisticated strategy within the proxemic group for coherent movement.

C. Intra-Group Proxemic Avoidance: Follow Strategy

After computing the avoidance side for the entire group, we can achieve coherent navigation at the group level. The main issue is to keep the agents connected during the navigation. In order to simulate this trajectory behavior, we let each agent dynamically follow some other agents in the same group whenever possible. In this way, the members in a group will move along the same local path and will have the minimal risk for group mix-up. To achieve this, we first need to decide whether one agent should be a leader or a follower in the group, and if it is a follower, we need to determine whom it should follow. Suppose we are given an agent $a \in G$ and its goal position $\mathbf{g}_a = e^s_a$. For the agent a, we check whether there is some member $b \in G$ such that $\|\mathbf{p}_b - \mathbf{g}_a\| < \|\mathbf{p}_a - \mathbf{g}_a\|$ and $\text{conn}(a, b) = 1$. In the other words, b can be connected to a and its position is closer to $\mathbf{g}_a$ than a. If so, then a will be a follower; otherwise it would be a leader. One example of followers and leaders is shown in Figure 4b.

If the agent a is a follower, we choose its following target as follows. First, we find all the agents b in the group that satisfy $\|\mathbf{p}_b - \mathbf{g}_a\| < \|\mathbf{p}_a - \mathbf{g}_a\|$ and $\text{conn}(a, b) = 1$, and the set of all qualified agents is denoted as F. In order to compute a stable connected group, we choose a's following target as one agent in F that is closest to a. If b is too far away from a then when a tries to follow b, the group shape may change. This will make it hard to perform group reciprocal avoidance. Formally, a's following target can be selected as: $b^* = \arg\min_{b \in F, b \neq a} \|\mathbf{p}_b - \mathbf{p}_a\|$. After selecting the following agent b, the new preferred velocity $\mathbf{v}^{\text{adapt}}_a$ for a is set along the direction $\mathbf{p}_b - \mathbf{p}_a$, which is used by ORCA constraints for local collision avoidance. A following example is shown between the agents a and b in Figure 4b. If the agent is a leader (e.g., the agent c in Figure 4b), its preferred velocity is set to its coherent avoidance velocity.

D. Avoiding Collisions between Agents

The adapted preferred velocity computed is used as the input to the ORCA agent-agent collision avoidance module. The ORCA algorithm computes the agents actual velocity and makes sure the agent avoids collisions with nearby individual agents. The agent need only avoid pairwise collisions with immediately neighboring agents. This cycle repeats indefinitely, and is carried out by each agent individually.

V. Implementation and Performance

In this section, we describe our implementation and highlight the performance of our navigation algorithm on several challenging benchmarks. We first adapt each agent's preferred velocity to simulate the proxemic behavior at the group-level, and then use RVO-based agent-agent collision avoidance to compute the actual velocity of each agent. We use two variants of RVO-based algorithms: ORCA [10] and HRVO [25]. We have implemented our algorithms in C++ on an Intel Core i7 CPU running at 3.30GHz with 16GB of RAM on Windows 7.

Method	Benchmark 1			Benchmark 2			Benchmark 3		
	tpf	#steps	#colls	tpf	#steps	#colls	tpf	#steps	#colls
HRVO	6.1	∞	∞	5.8	476	403	9.5	538	327
ORCA	5.2	∞	∞	6.2	399	442	8.7	460	287
Meso-scale	6.9	∞	∞	6.5	489	361	10.2	478	362
Ours + HRVO	6.5	357	12	6.9	187	0	10.9	401	0
Ours + ORCA	6.2	373	8	7.2	213	1	11.2	415	1

TABLE I: The comparison between our approach and previous methods on three benchmarks from three aspects: the tpf (running time per frame, in ms), the number of steps taken for all agents to reach the goal, and the number of agent-agent collisions during the navigation. The ∞ in # steps and # collisions means that some agents are not able to reach their goal during the simulation.

We use three different benchmarks to evaluate the performance. The first benchmark uses holonomic agents, while the other two benchmarks use car-like non-holonomic agents. For each benchmark, we compare the navigation results of our algorithms with prior multi-agent navigation algorithms: individual-agent based algorithms, ORCA [10] and HRVO [25], and one group-level navigation approach, the meso-scale ORCA [7]. The trajectories generated by the different methods are compared using three criteria: the actual simulation time, the number of steps required when all agents reach the goal positions, and the number of agent-agent collisions that occur during multi-agent navigation. More results are available in our technical report [26] and the supplemental video.

A. Benchmark 1: Two Groups of Agents

In this scenario, as shown in Figure 5, two groups of holonomic agents are moving towards each other in a narrow lane. Since there is little space for navigation, previous multi-agent navigation approaches are unable to compute collision free paths in this scenario (as shown by the HRVO and ORCA algorithms in Table I and the first row in Figure 5). In contrast, our algorithm can easily compute smooth collision-free paths for all agents by taking into account the proxemic behaviors. Our method has a small (about 5% to 12%) computational overhead as compared to ORCA and can easily handle hundreds of agents at real-time frame rates. In addition, our method results in relatively fewer collisions during the navigation.

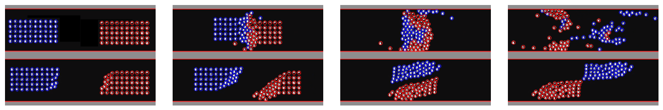
Fig. 5: In benchmark 1, two groups of agents are moving towards each other. The top row shows the trajectory behaviors' results using the ORCA-only algorithm, in which the agents do not exhibit the proxemic behavior. The bottom row highlights the performance of our algorithm that can generate coherent, proxemic behaviors and results in smooth trajectories.

B. Benchmark 2: Randomly Placed Non-holonomic Agents

In this benchmark, we generate two groups of non-holonomic agents with random initial positions that are moving towards each other in opposite directions, as shown in Figure 6. The results in Table I and Figure 6 for this benchmark show the ability of our method to generate a suitable number of groups in a decentralized manner. The agents start by forming two groups, and the right group automatically splits into two subgroups for more efficient navigation based on our optimization formulation. After that, the agents stay in their proxemic groups and reach the goal smoothly with at most one collision. In contrast, prior methods result in a large number of agent-agent collisions.

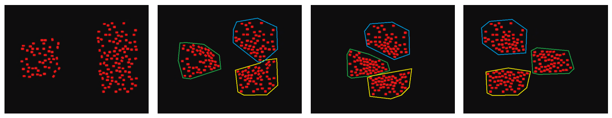

Fig. 6: In benchmark 2, the agents are randomly placed in two groups at the beginning and then move towards each other.

C. Benchmark 3: Multi-Group Agents in a Cluttered Scene

In this benchmark, we use a rectangular track as the environment and generate four groups of non-holonomic agents moving around the track. The results of our method are shown in Figure 7 and Table I. Each group uses one corner of the track as its goal. During the navigation, the agents dynamically adjust their velocity to stay inside the associated group. As compared with prior approaches, our method can generate smooth and coherent paths with at most one collision, and the computational time is comparable to previous methods.

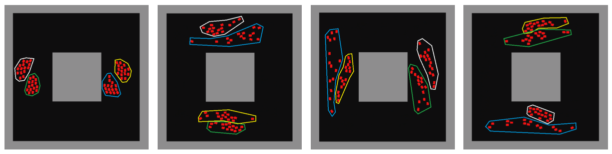

Fig. 7: In benchmark 3, four groups of agents are moving around a rectangular track. During the navigation, all agents exhibit proxemic group behaviors.

D. Trajectory Smoothness

We also compare the smoothness of the trajectories generated by ORCA and our approach on two different benchmarks for holonomic and non-holonomic agents respectively. As shown by the results in Figure 1, the trajectories provided by the ORCA algorithm will intertwine with each other, especially in the locations at which agents with different goals meet each other. Our approach leverages proxemic behavior to enable agents to avoid each other in a group manner and to generate smooth paths.

VI. Conclusions and Future Work

We present a novel multi-agent navigation algorithm that can automatically generate proxemic group behaviors. Our approach is general and makes no assumptions about group size or shape and can dynamically adapt to the environment. Moreover, it results in smooth and coherent navigation behaviors as compared to prior multi-agent reciprocal collision avoidance algorithms. We demonstrate its performance on complex benchmarks and highlight the benefits. The additional runtime overhead is $10 - 14\%$.

References

[1] S. LaValle and J. Kuffner, "Randomized kinodynamic planning," *IJRR*, vol. 20, no. 5, pp. 378–400, 2001.

[2] D. Fox, W. Burgard, H. Kruppa, and S. Thrun, "A probabilistic approach to collaborative multi-robot localization," *Autonomous Robots*, vol. 8, no. 3, pp. 325–344, 2000.

[3] R. Narain, A. Golas, S. Curtis, and M. C. Lin, "Aggregate dynamics for dense crowd simulation," *TOG*, vol. 28, no. 5, pp. 122:1–8, 2009.

[4] J. van den Berg, J. Snoeyink, M. Lin, and D. Manocha, "Centralized path planning for multiple robots: Optimal decoupling into sequential plans," in *RSS*, 2009.

[5] S. Bandini, A. Gorrini, L. Maneti, and G., "Crowd and pedestrian dynamics: Empirical investigation and simulation," in *Measuring Behavior*, 2012.

[6] O. B. Bayazit, J.-M. Lien, and N. M. Amato, "Better group behaviors in complex environments using global roadmaps," *Artificial Life 8*, vol. 8, p. 362, 2003.

[7] L. He and J. van den Berg, "Meso-scale planning for multi-agent navigation," in *ICRA*, 2013, pp. 2839–2844.

[8] I. Karamouzas and S. Guy, "Prioritized group navigation with formation velocity obstacles," in *ICRA*, 2015, pp. 5983–5989.

[9] I. Karamouzas and M. Overmars, "Simulating and evaluating the local behavior of small pedestrian groups," *TVCG*, vol. 18, no. 3, pp. 394–406, 2012.

[10] J. van den Berg, S. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics Research*, ser. Springer Tracts in Advanced Robotics, 2011, vol. 70, pp. 3–19.

[11] E. S. Knowles, "Boundaries around group interaction: The effect of group size and member status on boundary permeability." *Journal of Personality and Social Psychology*, vol. 26, no. 3, p. 327, 1973.

[12] L. Manenti, S. Manzoni, G. Vizzari, K. Ohtsuka, and K. Shimura, "An agent-based proxemic model for pedestrian and group dynamics: Motivations and first experiments," in *Multi-Agent-Based Simulation XII*, ser. Lecture Notes in Computer Science, 2012, vol. 7124, pp. 74–89.

[13] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," in *ICRA*, 1991, pp. 1398–1404 vol.2.

[14] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *RAM*, vol. 4, no. 1, pp. 23–33, 1997.

[15] D. Wilkie, J. van den Berg, and D. Manocha, "Generalized velocity obstacles," in *IROS*, 2009, pp. 5573–5578.

[16] D. Bareiss and J. van den Berg, "Generalized reciprocal collision avoidance," *IJRR*, 2015, to appear.

[17] I. Karamouzas, R. Geraerts, and A. van der Stappen, "Space-time group motion planning," in *Algorithmic Foundations of Robotics X*, ser. Springer Tracts in Advanced Robotics, E. Frazzoli, T. Lozano-Perez, N. Roy, and D. Rus, Eds., 2013, vol. 86, pp. 227–243.

[18] A. Best, S. Narang, S. Curtis, and D. Manocha, "Densesense: Interactive crowd simulation using density-dependent filters," in *SCA*, 2014.

[19] S. Curtis, J. Snape, and D. Manocha, "Way portals: Efficient multi-agent navigation with line-segment goals," in *I3D*, 2012, pp. 15–22.

[20] A. Krontiris, S. Louis, and K. Bekris, "Multi-level formation roadmaps for collision-free dynamic shape changes with non-holonomic teams," in *ICRA*, 2012, pp. 1570–1575.

[21] A. Kimmel, A. Dobson, and K. Bekris, "Maintaining team coherence under the velocity obstacle framework," in *AAMAS*, 2012, pp. 247–256.

[22] V. G. Santos and L. Chaimowicz, "Cohesion and segregation in swarm navigation," *Robotica*, vol. 32, pp. 209–223, 3 2014.

[23] A. Golas, R. Narain, S. Curtis, and M. C. Lin, "Hybrid long-range collision avoidance for crowd simulation," vol. 20, no. 7, 2014, pp. 1022–1034.

[24] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *IJRR*, vol. 17, no. 7, pp. 760–772, 1998.

[25] J. Snape, S. Guy, J. van den Berg, and D. Manocha, "Smooth coordination and navigation for multiple differential-drive robots," in *Experimental Robotics*, ser. Springer Tracts in Advanced Robotics, 2014, vol. 79, pp. 601–613.

[26] L. He, J. Pan, S. Narang, W. Wang, and D. Manocha, "Dynamic group behaviors for interactive crowd simulation," *CoRR*, vol. abs/1602.03623, 2015. [Online]. Available: http://arxiv.org/abs/1602.03623