FbCrowd: Interactive Multi-agent Simulation with Coupled Collision Avoidance and Human Motion Synthesis

Sahil Narang[*]
Universty of North Carolina Chapel Hill

Tanmay Randhavane[†]
University of North Carolina Chapel Hill

Andrew Best[‡]
University of North Carolina Chapel Hill

Dinesh Manocha[§]
University of North Carolina Chapel Hill

http://gamma.cs.unc.edu/pedvr

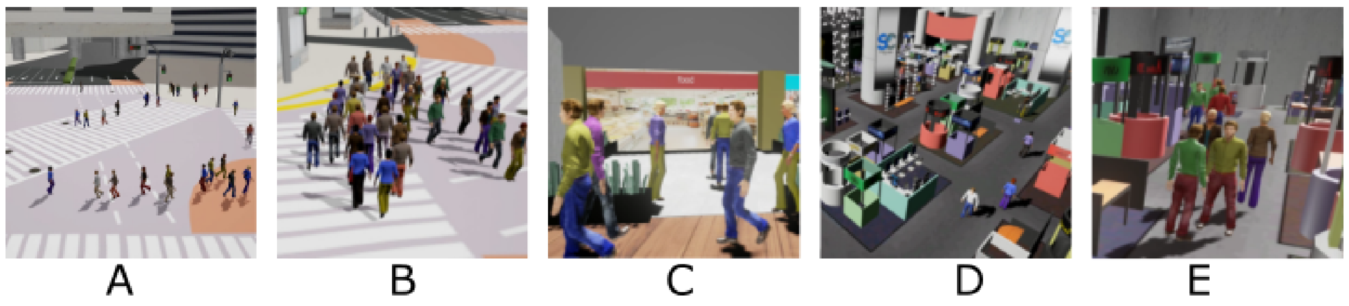

Figure 1: *Our full-body crowd simulation algorithm generates smooth and natural-looking collision-free trajectories for multiple agents at interactive rates using a coupled 2D planner and full-body motion synthesis. (A) A busy city crossing with multiple agents. (B) Agents show natural crowd behaviors such as lane formation. (C) A shopping mall scenario where agents pass each other in narrow hallways and walk up to shops. (D) A tradeshow scene where agents can be seen smoothly avoiding each other in narrow passages, and (E) even sidestepping in dense situations. Our algorithm can simulate and render 30 agents at 30-35 fps.*

Abstract

We present an interactive algorithm to generate plausible trajectories and full-body crowd simulations. Our formulation is based on a novel two-way coupling between 2D multi-agent collision avoidance and high-DOF human motion synthesis. We present a collision-free navigation algorithm that takes into account human motion and biomechanics constraints to compute smooth trajectories. Furthermore, we present a hybrid motion synthesis algorithm that seamlessly transitions between motion blending and semi-procedural locomotion, thereby balancing control and naturalness of the synthesized motion. The overall full-body crowd simulation algorithm can generate plausible motions with lower and upper body movements for multiple agents in dynamic virtual environments at interactive rates. We demonstrate its benefits over prior interactive crowd simulation algorithms.

1 Introduction

The problem of modeling realistic movement and behavior of multiple human-like characters is important in many applications, including computer animation, games, and CAD. One of the main challenges is to generate plausible simulations, in terms of visual and motion characteristics, for interactive applications such as virtual reality and games. The naturalness of the simulation is governed by the trajectory that each character chooses as well as the full-body animation of the walking character. Prior studies have concluded that many aspects of agent or pedestrian movement, including positions and orientations, are important for the realistic human perception of crowds [Ennis et al. 2011; Pelechano et al. 2008].

It is quite challenging to simulate a large group of human-like agents, especially in dense scenarios and in the presence of obstacles. Each human is an articulated character represented using many degrees of freedom. Hence, the total configuration space of a large crowd is very high-dimensional. Furthermore, no good techniques are known for modeling the dynamics of natural looking human motion in such high dimensions. As a result, most prior techniques decompose the crowd simulation problem into 2D navigation or path planning followed by 3D human motion animation. There is a large body of work [van den Berg et al. 2011; Treuille et al. 2006; Schadschneider 2002] that uses simple 2D representations for each agent (e.g., a disc) and computes collision-free trajectories in a 2D plane. Given the 2D trajectories of each agent, different methods [Welbergen et al. 2010] can be used as a post-process to generate walking animations along those trajectories. However, the 2D trajectory computation does not take into account human kinematic or dynamic stability constraints or full-body interactions in dense situations. There is some work on combining 2D navigation with full-body synthesis, but these methods are either too slow for interactive applications or may not generate natural-looking motions in some cases [Park et al. 2015; Shapiro 2011; Singh et al. 2011; Beacco et al. 2015].

Main Results: We present an interactive algorithm to generate plausible full-body movements of multiple humans in a shared space (i.e., full-body crowds or FbCrowd). Our approach accounts for several human motion constraints by incorporating feedback from full body motion synthesis into 2D trajectory computation. In order to generate collision-free, smooth and natural looking notions, we present three novel algorithms:

[*]sahil@cs.unc.edu
[†]tanmay@cs.unc.edu
[‡]best@cs.unc.edu
[§]dm@cs.unc.edu

- *Motion Constrained Navigation (MCN):* We present an efficient 2D multi-agent navigation algorithm that computes the 2D velocity for each agent based on constraints imposed by 3D human motion synthesis. These constraints are derived from captured human motion data-sets as well as biomechanics of human gait to generate human-like trajectories. The resulting 2D trajectories are natural looking with fewer artifacts as compared to prior 2D algorithms (Section 4).
- *Hybrid Human Motion Synthesis (HMS):* We use a hybrid scheme that seamlessly transitions between motion blending and semi-procedural locomotion based on the local environment. In contrast to prior methods, our approach dynamically balances control and naturalness of the synthesized motion (Section 5).
- *Two-way Coupling between Navigation and Synthesis (CNS):* There is a tight two-way coupling between the 2D multi-agent navigation and the 3D motion synthesis algorithm. Our coupling approach generates collision-free trajectories and plausible full body simulation (Section 3).

We combine these algorithms with a behavioral finite state machine (BFSM) to simulate complex behaviors in a number of indoor and outdoor scenes at interactive rates. Furthermore, we have integrated our system with the Unreal game engine to render the agents in real time. We demonstrate the benefits of our algorithm in many challenging scenarios and highlight the benefits over prior methods (Section 7). We also perform a preliminary user study to evaluate the benefits of our approach.

2 Related Work

In this section, we give a brief overview of prior work in multi-agent navigation and 3D human motion synthesis.

2.1 Interactive Multi-agent Navigation

Most prior 2D multi-agent techniques can be broadly classified as macroscopic models and microscopic models. Macroscopic models such as [Treuille et al. 2006] compute the aggregate motion of the agents by generating fields based on continuum theories of flows. Microscopic models, also called agent-based models, compute trajectories for each individual agent by decomposing the trajectory computation problem into two phases: global planning and local navigation. The global planners [LaValle 2006] compute a collision-free path through the environment considering only static obstacles. The local navigation algorithms [Helbing et al. 2000; Karamouzas et al. 2014; van den Berg et al. 2011; Ondřej et al. 2010; Singh et al. 2009; Stuvel et al. 2016; Bruneau and Pettré 2015] adapt the local motion of each agent to avoid collisions with dynamic obstacles and other agents. Some data-driven methods are capable of simulating virtual crowds with behaviors similar to captured crowd footage [Lee et al. 2007], or altering key properties to simulate varying crowd behaviors [Ju et al. 2010; Kwon et al. 2008]. Most of the above-mentioned methods use simple disc-based representations for each agent, except for [Stuvel et al. 2016] which employs capsule-shaped agents, and compute 2D trajectories based on those representations.

2.2 Interactive Human Motion Synthesis

There is extensive literature in computer graphics and animation on generating human like motion [Welbergen et al. 2010]. In this section, we limit our discussion to some data-driven, procedural, and physics-based methods that have been used for interactive applications.

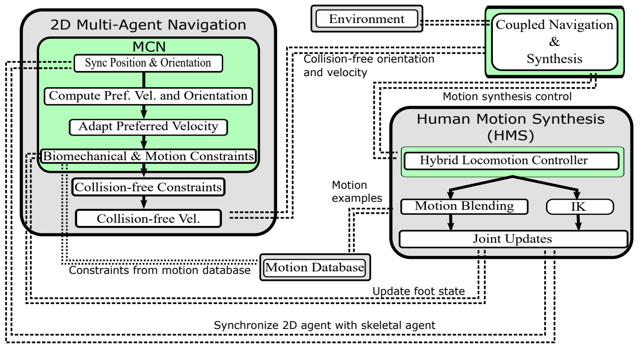
2D Multi-Agent Navigation
MCN
Sync Position & Orientation
Compute Pref. Vel. and Orientation
Adapt Preferred Velocity
Biomechanical & Motion Constraints
Collision-free Constraints
Collision-free Vel.
Environment
Coupled Navigation & Synthesis
Collision-free orientation and velocity
Motion synthesis control
Human Motion Synthesis (HMS)
Hybrid Locomotion Controller
Motion examples
Motion Blending
IK
Joint Updates
Motion Database
Constraints from motion database
Update foot state
Synchronize 2D agent with skeletal agent

Figure 2: ***Two-Way Coupling.*** *We highlight the two-way coupling between our 2D multi-agent navigation algorithm (MCN) and 3D human motion synthesis algorithms (HMS). The coupled navigation and synthesis algorithm (CNS) guides the motion synthesis based on the input from the 2D navigation algorithm and local environment, and is used to compute plausible motion at interactive rates.*

Data-driven methods can generate new trajectories by blending multiple motions. Motion graphs [Kovar et al. 2002; Feng et al. 2012; Min and Chai 2012] can generate a graph of motion clips and use search methods to compute a sequence of motion examples that are used to compute the desired locomotive trajectory. In general, these methods are limited by the underlying motion database, and often lack accurate control over the character for locomotion based applications. Some data-driven methods use spatio-temporal discretization to simulate multi-character interactions in large scenes [Lee et al. 2006; Shum et al. 2008; Kim et al. 2012; Won et al. 2014]. These methods are ideal for simulating interactions between few characters located within a fixed region (often referred to as tiles), with restricted movement between tiles. Choi et. al. [Choi et al. 2011] present a data-driven method for navigating complex static environments but do not account for agent-agent interactions.

Procedural methods generate locomotion by applying kinematic principles based on underlying biomechanics. These include inverted pendulum based models [Bruderlin and Calvert 1993], semi-procedural methods [Johansen 2009], physics-based approaches [Jain et al. 2009]. While such methods can compute physically correct motion, they may not always be natural looking.

2.3 Combining Multi-agent Navigation & Motion Synthesis

A few methods combine multi-agent navigation and motion synthesis to generate realistic motions for a large number of human-like agents. Some of these use a combination of robotics and physics-based simulation techniques [Park et al. 2015] which can be computationally expensive and may not generate human like motions. Other methods such as [Shapiro 2011; Singh et al. 2011; Beacco et al. 2015] are capable of real time simulations, but may not satisfy all the constraints.

3 Overview

In this section, we introduce the notation and terminology used in the rest of the paper. Furthermore, we give an overview of our two-way coupling algorithm.

3.1 Notation and Assumptions

We denote a scalar variable n with lower case letters, a vector $\mathbf{x}$ with a bold face lower case letter, a set $\mathcal{C}$ of entities with an upper case calligraphic letter. Each agent i in the simulator has an associated skeletal mesh, that is used for full-body motion synthesis. Each configuration $\mathbf{q}_i$ of the skeletal mesh is defined using the degrees-of-freedom (DOFs), including the 6-DOF root pose and the joint angles represented using n-dimensional vector space. We define the simulator state $\mathcal{S}$ as the union of all entities in the scene, including obstacles in the environment and the overall state space $\mathcal{Q} = \cup_i \mathbf{q}_i$.

We project the geometric representation of each skeletal mesh in $\mathbb{R}^n$ space to the $\mathbb{R}^2$ plane and bound it with a tightly fitted circle of radius r_i for multi-agent navigation. Therefore, each skeletal mesh with 6-DOF root joint $\mathbf{q}_i^{rt}$ is represented in the 2D multi-agent simulator by a circle of radius r_i positioned at $\mathbf{p}_i$, where $\mathbf{p}_i$ is simply the projection of the root joint $\mathbf{q}_i^{rt}$ on the 2D plane. The multi-agent navigation algorithm generates trajectories that correspond to the XY-projection of the 6-DOF root joint $\mathbf{q}_i^{rt}$ of the associated skeleton. These collision-free trajectories are represented as 2D time varying functions representing the position $\mathbf{p}_i(t)$ and velocity $\mathbf{v}_i(t)$.

Figure 2 highlights our approach and the various components. The main components are: the 2D navigation algorithm (MCN), the 3D human motion synthesis algorithm (HMS), and the two-way coupling between navigation and synthesis (CNS).

3.2 MCN: 2D Constraint Navigation Algorithm

We present a novel 2D navigation algorithm to compute smooth, collision-free trajectories. We use an agent-based approach, i.e., each agent is modeled as a discrete entity, represented as a 2D disc, with distinct goals, and computes its path independently of other agents. This path is represented by the instantaneous preferred velocity ($\mathbf{v}_i^{pref}$) i.e., the velocity in the direction of an immediate goal. Our MCN algorithm can be formally defined as a function $\mathbf{MCN}_i : \mathbb{S} \times \mathbb{R}^2 \times \mathbb{R} \rightarrow \mathbb{R}^2 \times \mathbb{R}$ to denote a function that maps the simulator state, the instantaneous preferred velocity, and time horizon, τ, into a collision-free 2D velocity, $\mathbf{v}_i$, with respect to other agents in the environment for at least time τ, and desired orientation o_i^d. Unlike prior navigation methods, we also take into account many motion constraints from the motion capture database as well as the skeletal mesh. This results in 2D trajectories that are amenable to full body motion synthesis and plausible simulation (Section 4).

3.3 HMS: 3D Human Motion Synthesis Algorithm

Our motion synthesis algorithm computes the trajectory $\mathbf{q}_i$ for the articulated skeleton in n-dimensional space. We present a hybrid locomotion algorithm that accounts for the mismatch in the dimensionality of the planning space for 2D navigation and full-body motion synthesis by dynamically balancing the naturalness of the motion and its fidelity with respect to the 2D trajectory (Section 5).

3.4 CNS: Two-way Coupling between Navigation and Synthesis

Ideally, the 2D collision-free velocity, $\mathbf{v}_i$ and the resulting trajectory, are precisely followed by the 3D motion synthesis algorithm. However, the high dimensionality of the motion synthesis algorithm tends to introduce some variability in the synthesised velocity of the root joint $\mathbf{q}_i^{rt}$. This implies that a collision-free 2D velocity computed by the navigation algorithm may still lead to collisions in the full-body synthesized motion. We overcome this drawback by tightly coupling the 2D navigation algorithm and the 3D motion synthesis algorithm.

First, we synchronize the position and orientation of the 2D disc with that of the root joint of the corresponding skeletal mesh. Second, we derive constraints from the motion database used for motion synthesis to limit the set of feasible velocities in the 2D planning stage. This imposes asymmetrical constraints in velocity space (Figure 3), similar to human movement. Overall, the 2D navigation algorithm guides the motion synthesis computation.

3.4.1 Hybrid Human Motion Synthesis

We prioritize motion-blending-based synthesis over semi-procedural synthesis unless there is a possibility of collision. At run-time, we compute the minimum time to collision [Karamouzas et al. 2014], ttc_i, and the local density [Narang et al. 2015], d_i, for every agent i in the simulation with respect to close-by agents and obstacles. Let t^{trans} and d^{thresh} denote user defined thresholds for time and local density respectively. We use the following condition to check if the character should use motion-blending-based synthesis:

$$ttc_i \leq t^{trans} \wedge d_i \leq d^{thresh}. \tag{1}$$

For other cases, the character should use semi-procedural synthesis to follow the 2D trajectory and minimize collisions. We choose t^{trans} such that it exceeds the maximum time to transition from one synthesis algorithm to the other. Moverover, we ensure that the planning horizon for the MCN algorithm exceeds t^{trans} so that the agent has enough time to transition into semi-procedural synthesis before any potential collisions. The actual transition mechanism is described in Section 5.

4 MCN : Motion Constrained Navigation

In this section, we present our novel 2D navigation algorithm that computes the 2D trajectory of each agent using a combination of global path planning and local navigation techniques. In contrast to prior approaches, our algorithm takes into account the current state of the skeletal mesh and many constraints related to full-body motion to generate 2D trajectories that can lead to natural looking motion synthesis.

4.1 Human Motion Constraints from Captured Data

We account for human kinematic and dynamic stability constraints by analyzing a database of human motions (Section 6). The motion database is comprehensive and consists of a wide range of human motions, implying that we can constrain motions that lie outside the set of motion examples. Each motion, $\mathbf{m}$, is parametrized in a $3D$ space defined by the scalar speed v^f, the turning rate ω^t, and the strafing rate ω^s. We begin by first mapping the motion examples, where $\omega^s = 0$, to velocity space. For example, the motion $\mathbf{m} = \{v^f, \omega^t, 0\}$ can be mapped to the 2D velocity $\mathbf{v}^{motion} = \{v^x, v^y\}$ as:

$$\mathbf{u} = \{cos\omega^t, sin\omega^t\} \tag{2}$$

$$\mathbf{v}^{motion} = \frac{\mathbf{u}}{\|\mathbf{u}\|}.v^f. \tag{3}$$

We wish to limit the set of feasible velocities to the space defined by the motion database by formulating half-plane constraints which can be efficiently solved. Figure 3(a) shows the motion examples in the velocity space, where each motion example is represented a point mapped according to Eq. 3. It can be seen that the wrapping polygon for the set of vertices is non-convex and thus, the

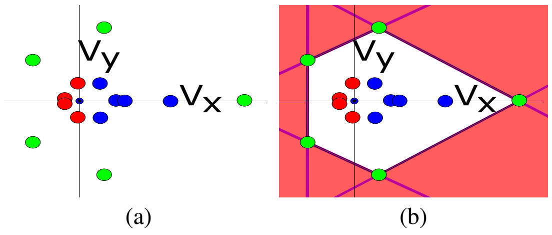

Figure 3: ***Motion Examples in the Velocity Space.*** *We choose velocities that lie inside the set of motion examples from a database. (a) Each vertex represents a motion from the database, visualized in the velocity space. (b) We build half-plane velocity constraints for the vertices lying on the convex hull (green). To avoid culling feasible velocities, we dynamically add constraints corresponding to sharp turning motions (red).*

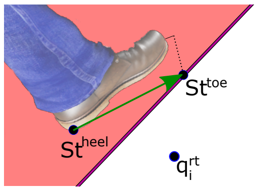

Figure 4: ***Biomechanical Constraints.*** *We use the full body motion synthesis algorithm to determine the position and orientation of the stance leg,* **st***, and constrain the movement of the swing leg to account for dynamic gait stability and to prevent self-collisions. The heel (*$\mathbf{st}^{heel}$*) and toe (*$\mathbf{st}^{toe}$*) of the stance foot (left) are used to construct a half-plane (red) of excluded velocities for the next planning step. This imposes human motion constraints on the 2D navigation algorithm, and generates 2D trajectories that are suitable for full body motion synthesis.*

corresponding half-plane constraints will cull velocities supported by the database. We overcome this by first computing a convex hull of the set of motion examples to yield a clockwise ordered set of n vertices $\mathcal{V} = \{\mathbf{v}_0^{motion}, \mathbf{v}_1^{motion}, ..., \mathbf{v}_{n-1}^{motion}\}$. Next, we compute half plane constraints for each edge of the convex hull (Figure 3(b)) and denote the set by $\mathcal{C}^{motion}$. For two consecutive vertices's $\mathbf{v}_i^{motion} = \{v_i^x, v_i^y\}$ and $\mathbf{v}_{i+1}^{motion} = \{v_{i+1}^x, v_{i+1}^y\}$, the half-plane constraint, C_i^τ, can be defined by the point $\mathbf{p}$ and direction vector $\mathbf{d}$ given as:

$$\mathbf{p} = \mathbf{v}_i^{motion}, \tag{4}$$

$$\mathbf{d} = \frac{\mathbf{v}_i^{motion} - \mathbf{v}_{i+1}^{motion}}{\|\mathbf{v}_i^{motion} - \mathbf{v}_{i+1}^{motion}\|}. \tag{5}$$

By considering the convex hull, we have included feasible velocities that are not contained within the motion database. This would also include a wide space of velocities where the character is turning behind i.e., $|\omega^t| > 90$. We dynamically add half-plane constraints if the preferred velocity, $\mathbf{v}^{pref}$, suggests a turn of more than 90 $^\circ$ from the current orientation of the character. This corresponds to the asymmetry in human motion i.e., turning motion is more restrictive than forward motion.

4.2 Biomechanical Constraints

The human walking gait cycle can be divided into six distinct periods that comprise the stance and swing phase [Vaughan et al. 1992]. For a gait cycle starting with the right foot leaving the ground, the first three phases i.e., initial double support, single limb stance, and second double support comprise the left stance phase while the next three phases i.e., initial swing, mid swing and terminal swing comprise the left swing phase. Previous studies in dynamic gait stability [Hof et al. 2005] have shown that center of mass of the character must stay over the base of support. It is likely that this condition is violated if the swing leg crosses the half-plane defined by the orientation and position of the stance leg. Moreover, the swing leg may also collide with the stance leg. We account for these constraints in our 2D navigation algorithm.

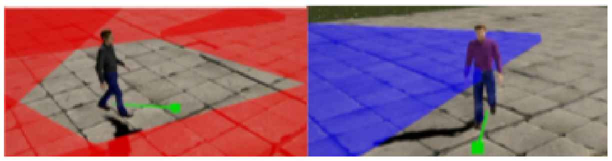

Figure 5: ***Human Motion Constraints.*** *Our 2D navigation algorithm (MCN) takes into account human motion constraints and formulates them as half-plane velocity constraints. (L) Motion constraints (i.e. the red region) derived from a database of motions. (R) Biomechanical constraints limit the set of feasible foot plants of the swing leg. Combined together, these motion constraints restrict the movement of the character based on the current full body pose and result in natural-looking synthesized motion.*

We use the full-body motion synthesis algorithm to determine the stance leg, St, and swing leg, Sw, of the character at every timestep. Let $\mathbf{St}^h$, $\mathbf{St}^b$ and $\mathbf{St}^t$ denote the positions of heel, ball and toe joints, respectively, of the stance leg that is projected on the ground plane. We wish to limit the set of feasible landing positions for the swing foot to a half-plane, H_{st}, defined by the stance-foot orientation vector $\mathbf{St}^o = \mathbf{St}^t - \mathbf{St}^h$. Humans can also turn sharply by twisting the stance foot in place. We account for this by rotating the orientation vector ($\mathbf{St}^o$) by a pre-defined threshold. We define a half-plane, H_{bm}, for the stance leg as:

$$H_{bm} = \{\mathbf{p} | (\mathbf{p} - \mathbf{st}^b).\mathbf{n}^b \geq 0\}, \tag{6}$$

where $\mathbf{n}^b$ denotes the normal to the stance foot vector $\mathbf{o}^{st}$ at $\mathbf{St}^b$, outward with respect to the root position $\mathbf{q}^{rt}$. We define a line segment between two points given $\mathbf{l}^1$ and $\mathbf{l}^2$ given by:

$$\mathbf{l}^1 = \mathbf{St}^h * k \tag{7}$$

$$\mathbf{l}^2 = \mathbf{St}^h * k + \mathbf{St}^o, \tag{8}$$

where $k = 1$ to ensure that the line segment $< \mathbf{l}^1, \mathbf{l}^2 >$ lies outside the bounding disc of the agent. We construct a velocity constraint C^{bm} for the line segment $< \mathbf{l}^1, \mathbf{2}^1 >$. Finally, adding C^{bm} to the set of motion constraints derived from human motions (Section 4.1) yields the set of full body motion constraints $\mathcal{C}^{motion}$.

4.3 Adapting Preferred Velocity to Local Conditions

The global-local paradigm for multi-agent simulation is often used to simulate a large number of agents at interactive rates. However, the fact that the global plan is independent of local dynamic conditions can lead to artifacts in the local planning. This can lead to noisy trajectories, increased collisions, and undesirable agent behaviors.

We adapt the preferred velocity generated by the global planner, $\mathbf{v}_i^{pref}$, for agent i to local dynamic conditions by considering "social forces" [Helbing et al. 2000; Karamouzas et al. 2014]. The *adapted preferred velocity*, $\mathbf{v}_i^{op}$, for the agent with mass m_i is given by:

$$m_i \frac{d\mathbf{v}_i^{0p}}{dt} = m_i \frac{\mathbf{v}_i^{pref} - \mathbf{v}_i^{0p}}{\tau_0} + \sum_{j \neq i} \mathbf{f}_{ij} + \sum_{W} \mathbf{f}_{iW}, \tag{9}$$

where $\mathbf{f}_{ij}$ and $\mathbf{f}_{iW}$ denote the repulsive forces due to neighboring agent j and obstacle W respectively. We use the formulation given by [Karamouzas et al. 2014] to compute the repulsive forces.We also scale down the preferred speed based on local density conditions [Narang et al. 2015], which can result in smoother trajectories.

4.4 Collision-free Velocity Computation

The social-forces model [Karamouzas et al. 2014] is effective at influencing the agent's plan w.r.t. local conditions, but is prone to collisions. We use reciprocal velocity obstacle [van den Berg et al. 2011] to formulate collision avoidance constraints $\mathcal{C}_i^{collision_\tau}$ for the planning time τ. The intersection of half-plane constraints $(\mathcal{C}_i^{collision_\tau} \cup \mathcal{C}_i^{motion})$, yields the set of feasible velocities for agent i. Similar to [van den Berg et al. 2011], we use linear programming to find a new collision-free 2D velocity $\mathbf{v}_i$ from this set that minimizes the deviation from the adapted preferred velocity.

4.5 Preferred Orientation

The 2D planner sets the desired forward facing vector $\mathbf{f}_i^d$ for agent i as:

$$\mathbf{f}_i^d = \begin{cases} \frac{\mathbf{v}_i^{pref}}{\|\mathbf{v}_i^{pref}\|}, & \text{if } \frac{\mathbf{v}_i^{pref}}{\|\mathbf{v}_i^{pref}\|} \cdot \frac{\mathbf{v}_i}{\|\mathbf{v}_i\|} \geq 0, t_i^{strafe} < t^{strafeLim} \\ \frac{\mathbf{v}_i}{\|\mathbf{v}_i\|}, & \text{otherwise} \end{cases}$$

where $\mathbf{v}_i^{pref}$ denotes the initial preferred velocity, $\mathbf{v}_i$ is the collision-free velocity. This formulation yields lateral movement, also called *strafing*, when $\mathbf{f}_i^d.\mathbf{v}_i \neq 0$. We track the contiguous time that the agent has been strafing t_i^{strafe} and limit it to a predefined threshold $t^{strafeLim}$. Finally, we set the desired orientation o_i^d to the angular representation of the unit vector $\mathbf{f}_i^d$.

5 HMS: Hybrid Human Motion Synthesis

An ideal motion synthesis algorithm should generate natural looking collision-free motion while precisely following the input 2D trajectory. However, existing synthesis methods tend to choose between the control or naturalness of the synthesized motion. Instead, we present a hybrid human motion synthesis algorithm (HMS) that generates full body motion to follow the 2D velocity computed by the MCN algorithm and balances control as well as naturalness of the full-body motion.

Our method seamlessly transitions between two widely used character locomotion techniques based on local dynamic conditions. We use a motion-blending technique [Feng et al. 2012] to generate natural looking motion. However, the synthesized motion may not precisely follow the desired 2D velocity. Thus, we transition to a semi-procedural technique [Juarez-Perez et al. 2014] in cases where the control over the character is critical. We present the algorithm to seamlessly transition between the techniques below and describe the specific conditions for such a transition in Section 3.4.1.

Let $\mathcal{M}_{mb} = \{m_1, m_2, ...m_n\}$ denote a set of motion clips used by the motion blending algorithm. We first identify the motion clip $m_{mb} \in \mathcal{M}_{mb}$, which is most similar to the motion clip used by semi-procedural locomotion m_{sp}, in terms of average walking speed s, turning angle ω^t, and strafing angle ω^s. Next, we build a set of correspondence times, $\mathcal{T} = \{(t_{sp}^1, t_{mb}^1), (t_{sp}^2, t_{mb}^2), ...\}$, which contain pairs of key times in the motion clips m_{sp} and m_{mb} at which the character poses are similar. We compute the set $\mathcal{T}$ offline and use it for transitions between the locomotion algorithms described below.

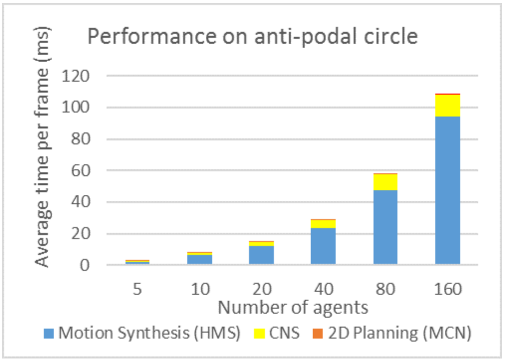

Figure 6: ***Performance Graph.*** *This graph shows the performance of our algorithm on the anti-podal circle with increasing numbers of agents. The motion synthesis module (HMS) dominates the overall computation time, compared to the 2D navigation algorithm (MCN) and the coupling algorithms (CNS). Our system can simulate* $40+$ *agents at* 30 *fps, excluding rendering costs.*

5.1 Transition to Semi-procedural Locomotion

We begin by smoothly manipulating the blending weights such that the blending algorithm only uses $m_{mb} \in \mathcal{M}_{mb}$. Once m_{mb} is in use, we use the pre-computed correspondence set $\mathcal{T}$ to find a suitable time to transition. Given the current time t_{mb} in the walk cycle m_{mb}, we compute the suitable correspondence pair $\mathcal{T}_i = (t_{sp}^i, t_{mb}^i)(\in \mathcal{T})$ based on the condition:

$$(t_{mb}^i - t_{mb} \geq 0) \wedge ((t_{mb}^i - t_{mb}) < (t_{mb}^j - t_{mb}) \forall_j t_{mb}^j > t_{mb}). \quad (10)$$

Once we have found a suitable correspondence pair, $\mathcal{T}_i = (t_{sp}^i, t_{mb}^i)(\in \mathcal{T})$, we initialize semi-procedural locomotion with time t_{sp}^i, when the current time t_{mb} equals or surpasses t_{mb}^i. We use a similar approach to transition to the motion blending algorithm.

6 Implementation and Performance

In this section, we present the implementation details of different components of our system. We also highlight the performance of our approach in different scenarios.

BFSM: We use a BFSM to represent the behavioral state of each agent in the simulation. The BFSM maps the time and simulator state into a goal position $\mathbf{g}_i$ for agent i. We utilize the crowd simulation framework Menge [Curtis et al. 2016] to implement our local navigation algorithm.

Global Path Planning: We employ a navigation mesh to plan a collision-free path with respect to static obstacles in the environment. The global planner maps the simulator state and the agent's goal position into a instantaneous preferred velocity, $\mathbf{v}_i^{pref}$, and preferred orientation, o_i^d.

Motion Database: We leverage the motion database described in [Shapiro 2011] to generate the motion of each agent. The database comprises of 19 different locomotion examples.

6.1 Performance

We have implemented our algorithm in C++ on a Windows 10 desktop PC. All the timing results in the paper were generated on an Intel Xeon E5-1620 v3 with 4 cores and 16 GB of memory. Our current implementation is not optimized. We present the timing results (Figure 6) on the anti-podal circle benchmark where agents are placed on the circumference of the circle with diametrically opposite goals. In practice, the HMS algorithm is significantly more expensive than MCN, especially as the number of agents in the scene increases. Moreover, MCN is easier to parallelize on multiple cores. IOur system can generate the trajectories and full-body motion of many tens of agents at interactive rates on desktop PCs and has been integrated with the Unreal game engine that is used for rendering.

7 Results

We highlight the results of our approach on several challenging benchmarks and discuss benefits over prior approaches.

7.1 Benchmarks

We demonstrate the performance of our approach on three benchmark scenarios, shown in Figure 1.

Shibuya Crossing We simulate a busy street crossing (Figure 1(A-B)), where agents are probabilistically assigned goal positions and must use the pedestrian walk lanes to navigate. This forces the agents to constantly avoid collisions with other agents in the scene. Subtle collision avoidance behaviors can be seen (in the video), when the agents change their path to avoid collisions. In some cases, overt collision avoidance behaviors such as sidestepping movement can also be observed as well. Our system can simulate and render 30+ agents at approx. 30-35 fps.

Tradeshow We simulate a tradeshow scenario (Figure 1(c)) which is challenging due to the high number of obstacles and narrow passages. Our approach heavily exploits the BFSM to simulate behaviors such as walking up to a randomly assigned booth and facing towards the booth for a few seconds. Agents can be seen smoothly avoiding collisions with one other in the narrow passages, forming lanes and and often sidestepping to avoid each other (Figure 1(D-E)). Our system can simulate and render 50+ agents at 15-20 fps.

Shopping Mall This scenario shows a shopping mall (Figure 1(C)) where agents walk around the shops and pass each other in the narrow hallways. Overall, we observe smooth trajectories and collision avoidance behaviors. Our system can simulate 15 agents at 50-60 fps, including rendering cost.

Obstacle Course: Evaluating the Benefits of HMS
We evaluate the benefits of our HMS algorithm on a challenging scene with narrow and sharp turning passages. A purely motion blending approach generates smooth motion in open space but causes collisions in the tight corners. On the other hand, a purely semi-procedural approach generates collision-free trajectories but the resulting motion is prone to artifacts such as mechanical looking motions. In contrast, our hybrid motion synthesis algorithm generates natural looking motion in open space, and seamlessly transitions to the semi-procedural approach as the character approaches the narrow passageways, as can be seen in the video.

7.2 Comparisons

We have compared the performance of our approach with prior methods. These include comparisons between MCN and prior 2D navigation algorithms; and comparison of CNS with prior coupled crowd simulation algorithms and systems.

7.2.1 Decoupled 2D Navigation Algorithms

We couple the motion blending based synthesis algorithm with prior 2D navigation methods to evaluate the benefits of our 2D planning algorithm (MCN). We use the following benchmarks and present results in Table 1.

- **2-Way Crossflow:** In this benchmark, two populations, each with 15 agents, cross each other orthogonally. Agents with MCN slow down appropriately as they approach the congested intersection, sidestep and find gaps to avoid each other. Thus, MCN algorithm results in fewer collisions as compared to ORCA and SF. Furthermore, MCN generates smoother trajectories, indicated by the lower average acceleration value.
- **Bidirectional Flow:** In this benchmark, two groups of agents approach each other at an angle of $180°$. ORCA agents abruptly change velocities to avoid collisions leading to noisy trajectories. MCN agents attempt to smoothly navigate past each other which leads to slightly higher number of collisions. Compared to SF, both MCN and ORCA agents depict crowd behaviors such as lane formation.
- **4-Way Crossflow:** In this scenario, four groups of agents are initialized at the corners of a square with diagonally opposite goals. Agents with MCN often sidestep and execute tight turns to avoid each other. In contrast, the SF algorithm is unstable due to the high timestep ($t = 0.1$ s) which leads to significantly larger number of collisions. Furthermore, the MCN algorithm generates smoother trajectories compared to both ORCA and SF.

These benchmarks demonstrate that MCN results in fewer collisions, smoother trajectories and more stable behaviors as compared to prior multi-agent navigation, even at high time-steps. Furthermore, MCN can automatically generate many emerging behaviors including commonly observed crowd behaviors such as lane formations, arching at bottlenecks, etc.

7.2.2 Coupled Approaches

There are some prior interactive crowd simulation systems that provide some coupling between navigation and synthesis. Some of these use footstep-based planners. Singh et al. [2011] use an inverted pendulum model to generate a timed sequence of footsteps that can be followed precisely using procedural animation. Beaccho et. al. [2015] use motion interpolation and blending to synthesize motion which can follow the footstep trajectory. These footstep-based planners impose some biomechanical constraints, but also make some simplifying assumption that can impact the plausibility of the generated 2D trajectories and lead to artifacts in motion synthesis. Furthermore, the method presented by Beaccho et al. introduces a user-defined constant that prioritizes between fidelity of root movement and footstep placement in the synthesized motion. In contrast, our method uses local conditions to dynamically transition between two different motion synthesis algorithms, thus balancing control and naturalness of the synthesized motion. Park et al. [2015] use a coupled approach based on full-body motion planning. It is not fast enough for interactive applications and may not generate natural looking motion.

Shapiro et al. [2011] present a character animation approach that utilizes a 2D steering algorithm and a motion-blending-based technique to generate visually plausible motion. However, their method prioritizes naturalness of the synthesized motion and is prone to collisions in medium to high density crowds. We evaluated the smoothness of the trajectories generated by our algorithm (CNS), with those generated by Smartbody (using Steerlib) on the anti-podal circle benchmark with 17 agents. Figure 7 highlights the trajectory of each agent with a different color. The agents in our approach, CNS, are able to navigate to their goals faster with smoother trajectories. On the other hand, Smartbody (with Steerlib) can generate noisy trajectories with significantly higher

Benchmark	Num. Agents	Collisions			Average Acceleration			Average frame update time (ms)		
		ORCA	SF	MCN	ORCA	SF	MCN	ORCA	SF	MCN
2-way Crossflow	30	0.0566	0.1440	0.0496	0.1542	0.1253	0.1135	0.40	0.32	0.43
Bidirectional Flow	53	0.0491	0.0539	0.0722	0.1901	0.1683	0.1121	0.66	0.57	0.70
4-way Crossflow	100	0.2027	0.3369	0.1791	0.0532	0.0390	0.0387	6.40	6.53	17.71

Table 1: ***Comparing MCN with prior 2D Navigation Algorithms.*** *We evaluate our 2D planning algorithm, MCN, with prior methods based on velocity obstacles (ORCA) and social forces (SF). All three 2D navigation methods are coupled with motion blending based synthesis and simulated at a fixed time step. We compare (a) the number of agent-agent collisions, measured using interval penetration depth averaged over all frames and agents, (b) the average acceleration of the root joint over all frames and agents, and (c) the average frame update time for 2D planning. Our MCN algorithm accounts for the full body pose during 2D planning leading to fewer collisions in the synthesized motion. Moreover, it generates smoother trajectories, indicated by the relatively lower average acceleration, at a slightly higher run time cost.*

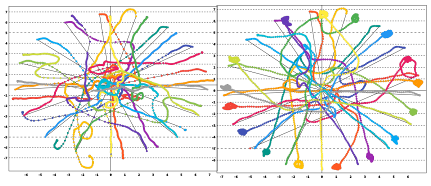
(a) Smartbody (b) CNS

Figure 7: ***Trajectory Comparisons on the Anti-podal circle benchmark.*** *We visualize the root joint position of each agent using a different color. (a) The agent trajectories generated in Smartbody (using steerlib) exhibit several collisions and noisy trajectories. (b) Our method, CNS, results in fewer collisions and smoother trajectories. Moreover, our agents reach their goals faster than Smartbody.*

number of collisions. This can be observed in the middle region near the center of the circle in Figure 7(b). These behaviors are also illustrated in the video.

There has been extensive work on synthesizing natural looking interactions between virtual characters. Many of these methods rely on spatial discretization that may not be suitable for locomotion based behaviors [Lee et al. 2006; Shum et al. 2008; Won et al. 2014; Hyun et al. 2013], or do not provide collision avoidance guarantees for multi-agent navigation in dynamic environments [Kim et al. 2009; Choi et al. 2011]. In contrast, our approach is more suitable for navigation-based behaviors and provides collision-free trajectory computations for tens of agents in dense environments. Also, our multi-agent navigation algorithm (MCN) can be easily integrated with other motion synthesis algorithms [Min and Chai 2012] and animation systems such as Morpheme or Unity's Mecanim.

7.3 User Evaluation

We conducted a within-subjects user study to evaluate the benefits of our coupled navigation and synthesis algorithm (CNS) as compared to a decoupled method. In case of the decoupled method, we used ORCA to first generate 2D trajectories and then a motion blending based synthesis method which exactly followed the 2D trajectory. The study comprised of two scenes: anti-podal circle with 17 agents, and bidirectional flow with 18 agents. For each scene, the user was presented with a pair of motion clips, one simulated with CNS and the other with a decoupled planner. We asked the users to rate the clips using a 7 point Likert scale with values labeled (Left Much Better, Left Better, Left Slightly Better, No Difference, Right Slightly Better, Right Better, Right Much Better). In this response format, a value of one indicates a strong preference for the clip listed on the left of the presentation. The left and right order of presentation, as well as that of the scenes was counterbalanced. The user were asked to rate the pair of motion clips on the questions of "naturalness of the motion", "naturalness of crowd interactions","amount of collisions", and "amount of artifacts".

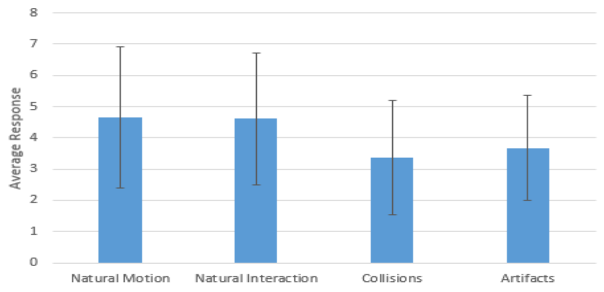

Figure 8: ***User Responses in the Antipodal Circle scenario.*** *For each question, participants rated their preference on a scale of 1 to 7 with 7 representing the highest preference for our system. Users prefer our approach to a decoupled approach in 62% of the responses when asked to evaluate the naturalness of motion. In particular, 33% of the users gave our method the highest possible rating. Similar responses were observed on the question of naturalness of the crowd interaction, with 61.9% preferring our method.*

The user study was taken by 21 participants on an online portal. In case of the anti-podal circle scenario, the user responses showed a preference for our approach with 62% favouring our coupled method on the question of naturalness of motion, as compared to the decoupled method (Figure 8). In particular 33% of the users gave our method the highest possible rating, compared to 14.3% for the decoupled method. Similar responses were observed on the question of naturalness of the crowd interaction, with 61.9% preferring our method compared to 23.8% for the decoupled planner. Of these responses, 52.3% indicated strong or very strong preference (6 or 7), compared to 19.05% for the decoupled planner. In the bidirectional flow scenario, our method was preferred to the decoupled method in 33% of responses on the question of naturalness of motion and 19% on the question of naturalness of interaction. We attribute this discrepancy to the simplicity of the computed paths and relatively low average densities, creating less adversarial conditions.

8 Conclusion, Limitations & Future Work

We present an interactive approach for full-body crowd simulation in a shared space. Our formulation computes collision-free trajectories and plausible full body motions for each agent. We present a novel two-way coupling between 2D navigation and 3D human motion synthesis, along with a constrained 2D navigation and a hybrid 3D human motion synthesis algorithm. We have demonstrated the interactive performance of our overall algorithm in many scenarios and highlight the benefits over prior crowd simulation methods.

Our approach has some limitations. Given the overall goal of interactive performance, our human motion synthesis algorithm is a hybrid combination of motion blending and inverse kinematics.

It may be possible to generate more natural looking motions using data-driven or physics-based simulation algorithms, but they tend to be more expensive. Our coupled method does not take into account many behaviors, biomechanical, and other natural-looking constraints. The range of movements is limited by the number of locomotion examples in the motion database. Most current motion databases consist of humans walking in open spaces, and may not capture full body motions corresponding to pairwise interactions between the agents in the database.

There are many avenues for future work. Besides overcoming the limitations, we would like to improve the fidelity as well as performance of the hybrid human motion synthesis algorithm. It could be useful to evaluate the benefits of the movements generated by our algorithm in terms of realistic human perception of crowds, adding different gestures [Ennis et al. 2011; Pelechano et al. 2008; Narang et al. 2016], and also motion styles based on high level attributes such as personality [Durupinar et al. 2016]. Our MCN and CNS can also be combined with other human motion synthesis algorithms [Lee et al. 2006; Shum et al. 2008; Kim et al. 2012; Won et al. 2014]

References

BEACCO, A., PELECHANO, N., KAPADIA, M., AND BADLER, N. I. 2015. Footstep parameterized motion blending using barycentric coordinates. *Computers & Graphics 47*, 105–112.

BRUDERLIN, A., AND CALVERT, T. 1993. Interactive animation of personalized human locomotion. In *Proc. of Graphics Interface*, 17–23.

BRUNEAU, J., AND PETTRÉ, J. 2015. Energy-efficient mid-term strategies for collision avoidance in crowd simulation. In *Proceedings of the 14th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 119–127.

CHOI, M. G., KIM, M., HYUN, K. L., AND LEE, J. 2011. Deformable motion: Squeezing into cluttered environments. *Computer Graphics Forum 30*, 2, 445–453.

CURTIS, S., BEST, A., AND MANOCHA, D. 2016. Menge: A modular framework for simulating crowd movement. *Collective Dynamics 1*, 1–40.

DURUPINAR, F., KAPADIA, M., DEUTSCH, S., NEFF, M., AND BADLER, N. I. 2016. Perform: Perceptual approach for adding ocean personality to human motion using laban movement analysis. *ACM Trans. Graph. 36*, 1, 6:1–6:16.

ENNIS, C., PETERS, C., AND O'SULLIVAN, C. 2011. Perceptual effects of scene context and viewpoint for virtual pedestrian crowds. *ACM Transactions on Applied Perception (TAP) 8*, 10.

FENG, A. W., XU, Y., AND SHAPIRO, A. 2012. An example-based motion synthesis technique for locomotion and object manipulation. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, ACM, 95–102.

HELBING, D., FARKAS, I., AND VICSEK, T. 2000. Simulating dynamical features of escape panic. *Nature 407*, 487–490.

HOF, A., GAZENDAM, M., AND SINKE, W. 2005. The condition for dynamic stability. *Journal of biomechanics 38*, 1, 1–8.

HYUN, K., KIM, M., HWANG, Y., AND LEE, J. 2013. Tiling motion patches. *IEEE Trans. Vis. Comput. Graph 19*, 11.

JAIN, S., YE, Y., AND LIU, C. K. 2009. Optimization-based interactive motion synthesis. *ACM Trans. Graph. 28*, 1 (Feb.), 10:1–10:12.

JOHANSEN, R. S. 2009. *Automated semi-procedural animation for character locomotion*. PhD thesis, Aarhus Universitet, Institut for Informations-og Medievidenskab.

JU, E., CHOI, M. G., PARK, M., LEE, J., LEE, K. H., AND TAKAHASHI, S. 2010. Morphable crowds. *ACM Trans. Graph. 29*, 6 (Dec.), 140:1–140:10.

JUAREZ-PEREZ, A., FENG, A., KALLMANN, M., AND SHAPIRO, A. 2014. Deformation, parameterization and analysis of a single locomotion cycle. In *Proceedings of the Seventh International Conference on Motion in Games*, 182–182.

KARAMOUZAS, I., SKINNER, B., AND GUY, S. J. 2014. Universal power law governing pedestrian interactions. *Physical Review Letters 113*, 23, 238701.

KIM, M., HYUN, K., KIM, J., AND LEE, J. 2009. Synchronized multi-character motion editing. In *ACM Transactions on Graphics (TOG)*, vol. 28, 79.

KIM, M., HWANG, Y., HYUN, K., AND LEE, J. 2012. Tiling motion patches. In *Proceedings of the 11th ACM SIGGRAPH/Eurographics conference on Computer Animation*, 117–126.

KOVAR, L., GLEICHER, M., AND PIGHIN, F. 2002. Motion graphs. In *ACM Transactions on Graphics (TOG)*, vol. 21, 473–482.

KWON, T., LEE, K. H., LEE, J., AND TAKAHASHI, S. 2008. Group motion editing. In *ACM Transactions on Graphics (TOG)*, vol. 27, 80.

LAVALLE, S. 2006. *Planning Algorithms*. Cambridge.

LEE, K. H., CHOI, M. G., AND LEE, J. 2006. Motion patches: building blocks for virtual environments annotated with motion data. In *ACM Transactions on Graphics (TOG)*, vol. 25, 898–906.

LEE, K. H., CHOI, M. G., HONG, Q., AND LEE, J. 2007. Group behavior from video: a data-driven approach to crowd simulation. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 109–118.

MIN, J., AND CHAI, J. 2012. Motion graphs++: a compact generative model for semantic motion analysis and synthesis. *ACM Transactions on Graphics (TOG) 31*, 6, 153.

NARANG, S., BEST, A., CURTIS, S., AND MANOCHA, D. 2015. Generating pedestrian trajectories consistent with the fundamental diagram based on physiological and psychological factors. *PLoS ONE 10*, 4 (04), 1–17.

NARANG, S., BEST, A., RANDHAVANE, T., SHAPIRO, A., AND MANOCHA, D. 2016. PedVR: Simulating gaze-based interactions between a real user and virtual crowds. *Proc. of ACM VRST*.

ONDŘEJ, J., PETTRÉ, J., OLIVIER, A.-H., AND DONIKIAN, S. 2010. A synthetic-vision based steering approach for crowd simulation. In *Proc. SIGGRAPH*, 123:1–123:9.

PARK, C., BEST, A., NARANG, S., AND MANOCHA, D. 2015. Simulating high-dof human-like agents using hierarchical feedback planner. In *Proceedings of the 21st ACM Symposium on Virtual Reality Software and Technology*, ACM, 153–162.

PELECHANO, N., STOCKER, C., ALLBECK, J., AND BADLER, N. 2008. Being a part of the crowd: towards validating vr crowds using presence. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*, 136–142.

SCHADSCHNEIDER, A. 2002. Cellular automaton approach to pedestrian dynamics - theory. *Pedestrian and Evacuation Dynamics*, 75–86.

SHAPIRO, A. 2011. Building a character animation system. In *Motion in Games*, J. Allbeck and P. Faloutsos, Eds., vol. 7060 of *Lecture Notes in Computer Science*, 98–109.

SHUM, H. P., KOMURA, T., SHIRAISHI, M., AND YAMAZAKI, S. 2008. Interaction patches for multi-character animation. In *ACM Transactions on Graphics (TOG)*, vol. 27, 114.

SINGH, S., KAPADIA, M., FALOUTSOS, P., AND REINMAN, G. 2009. An open framework for developing, evaluating, and sharing steering algorithms. In *Proceedings of the 2nd International Workshop on Motion in Games*, 158–169.

SINGH, S., KAPADIA, M., REINMAN, G., AND FALOUTSOS, P. 2011. Footstep navigation for dynamic crowds. *Computer Animation and Virtual Worlds 22*, 2-3, 151–158.

STUVEL, S., MAGNENAT-THALMANN, N., THALMANN, D., VAN DER STAPPEN, A. F., AND EGGES, A. 2016. Torso crowds. *IEEE Transactions on Visualization and Computer Graphics*.

TREUILLE, A., COOPER, S., AND POPOVIĆ, Z. 2006. Continuum crowds. In *Proc. of ACM SIGGRAPH*, 1160–1168.

VAN DEN BERG, J., GUY, S. J., LIN, M., AND MANOCHA, D. 2011. Reciprocal n-body collision avoidance. In *Inter. Symp. on Robotics Research*, 3–19.

VAUGHAN, C. L., DAVIS, B. L., AND O'CONNOR, J. C. 1992. *Dynamics of human gait*. Human Kinetics Publishers Champaign, Illinois.

WELBERGEN, V. H., BASTEN, V. B., EGGES, A., RUTTKAY, Z., AND OVERMARS, M. 2010. Real time character animation: A trade-off between naturalness and control. *Computer Graphics Forum 29*, 8.

WON, J., LEE, K., O'SULLIVAN, C., HODGINS, J. K., AND LEE, J. 2014. Generating and ranking diverse multi-character interactions. *ACM Transactions on Graphics (TOG) 33*, 6, 219.

Appendix

9 2D Collision-free Velocity Computation

In this section, we provide additional details of our novel 2D navigation algorithm, MCN. We begin by computing half-plane velocity constraints related to full body motion, as described in Sections 4.1 & 4.2 of the main document. We refer to these motion constraints as $\mathcal{C}^{motion}$.

9.1 Adapting Preferred Velocity to Local Conditions

We adapt the preferred velocity, $\mathbf{v}_i^{pref}$, to local dynamic conditions, as detailed in Section 4.3 of the main document. We use a social forces formulation [Karamouzas et al. 2014] followed by a density filter [Narang et al. 2015] to yield a more appropriate preferred velocity, referred to as the *adapted preferred velocity* $\mathbf{v}_i^{op}$.

The force $\mathbf{f}^{ij}$ experienced by pedestrian i due to the interaction with another pedestrian j is formulated as:

$$\mathbf{f}^{ij} = -\nabla_{\mathbf{p}_{ij}}(k\tau^{-2}e^{-\tau/\tau_0}), \tag{11}$$

where $\nabla_{\mathbf{p}_{ij}}$ is the spatial gradient, $\mathbf{p}_{ij} = \mathbf{p}_i - \mathbf{p}_j$ is the relative displacement of i and j, τ is the time to collision or interaction, k and τ_0 are constants. A similar formulation is used to compute the repulsive force $\mathbf{f}^{iW}$ for every neighboring obstacle W. The adapted preferred velocity $\mathbf{v}_i^{op}$ can then be computed using Eq. 9 of the main document. We use the same values for the constants as described in [Karamouzas et al. 2014]. Finally, we scale down the preferred speed based on local density conditions which can result in smoother trajectories.

The main advantage of such a model is that each agents plan is in- fluenced by the neighboring agents. Furthermore, this influence depends on the local conditions in the environment. For example. two agents moving towards each other at a small relative velocity should influence each other less as compared to two agents that approach each other at a high relative velocity. However, the exponential response function introduces issues with numeric stability, wherein the forces acting on the agents can cause jittery behavior i.e. high frequency oscillations in velocities, especially in dense scenarios. In general, combining these forces to guarantee collision avoidance is an open problem. These issues are generally avoided by significantly reducing the simulation timestep to a small value, e.g., 0.001 seconds, which can slowdown the overall simulation. However, using such a low timestep may not be possible for interactive applications. We overcome this issue by using the social forces model to only modulate the preferred velocity and impose additional constraints to select a collision-free 2D velocity, as described below.

9.2 Collision-free constraints

The social-forces model is effective at influencing the agent's plan w.r.t. local conditions but are prone to collisions. We use the reciprocal velocity obstacle (ORCA) based constraints [van den Berg et al. 2011] to enforce collision avoidance. For each neighboring agent j, moving with current velocity $\mathbf{v}_j$, we compute the half plane constraint C_{ij}^{τ}. Effectively, the constraint C_{ij}^{τ} represents a half plane of collision-free velocities for agent i with respect to agent j for the planning time τ. Similarly, we build a half plane constraint C_{iW}^{τ} for each nearby obstacle W. We refer to these constraints as *Collision-free Constraints*, $\mathcal{C}_i^{collision_\tau}$, defined as:

$$\mathcal{C}_i^{collision_\tau} = \cup_j C_{ij}^{\tau} \cup_W C_{iW}^{\tau} \tag{12}$$

9.3 Collision-free Velocity Computation

At every time step, the algorithm computes the preferred velocity $\mathbf{v}_i^{pref}$, adapts it to local conditions to yield $\mathbf{v}_i^{op}$, and generates half-plane human motion constraints, $\mathcal{C}_i^{motion}$, for each agent i in the simulation. Next, we compute the collision-avoidance constraints, $\mathcal{C}_i^{u_\tau}$.

$$\mathcal{C}_i^{total} = \mathcal{C}_i^{collision_\tau} \cup \mathcal{C}_i^{motion}. \tag{13}$$

The intersection of all the half-plane constraints yields the convex set MCN_i^{τ} of collision-free velocities that respect motion constraints. Similar to [van den Berg et al. 2011], we use linear programming to find a new collision-free 2D velocity $\mathbf{v}_i$ from the set MCN_i^{τ} that minimizes the deviation from the adapted preferred velocity.

$$\mathbf{v}_i = \min_{\mathbf{v}_i \in MCN_i^{\tau}} \|\mathbf{v}_i - \mathbf{v}_i^{op}\| \tag{14}$$

9.4 Dense Conditions

In dense conditions, there might be instances where the 2D linear program fails to find a solution because MCN_i^{τ} is empty. In such cases, we choose the "safest possible" velocity for the agent, i.e. the velocity that minimally penetrates the constraints induced by the other agents. This can be done by solving a three dimensional linear program, where the signed distance to the half plane represents the third dimension, as described in [van den Berg et al. 2011]. However, in contrast to ORCA, we prioritize agents in order of the time to collision which reduces the number of collisions. Thus, an agent prioritizes collision avoidance with a agent heading towards it as opposed to a closer agent heading away.

10 Transition to Motion-blending Based Locomotion

We employ a hybrid motion synthesis algorithm that seamlessly transitions between motion blending and semi-procedural synthesis based on local dynamic conditions. The algorithm for transitioning to semi-procedural locomotion is described in Section 5.1 of the main document. Here, we provide details on mechanism for transitioning to motion blending based synthesis.

Let $\mathcal{M}_{mb} = \{m_1, m_2, ...m_n\}$ denote a set of motion clips used by the motion blending algorithm. Moreover, let $m_{mb} \in \mathcal{M}_{mb}$ denote the motion clip that is most similar to the motion clip used by semi-procedural locomotion m_{sp}, in terms of average walking speed s, turning angle ω^t, and strafing angle ω^s. Also, let $\mathcal{T} = \{(t_{sp}^1, t_{mb}^1), (t_{sp}^2, t_{mb}^2), ...\}$, denote the set of pairs of key times in the motion clips m_{sp} and m_{mb}, for which the character poses are most similar.

Given the current time in walk cycle for semi-procedural locomotion t_{sp}, we find the suitable correspondence pair $\mathcal{T}_i = (t_{sp}^i, t_{mb}^i)(\in \mathcal{T})$ as follows:

$$(t_{sp}^i - t_{sp} \geq 0) \wedge ((t_{sp}^i - t_{sp}) < (t_{sp}^j - t_{sp}) \forall_j t_{sp}^j > t_{sp}). \tag{15}$$

When the current time in the walk cycle for semi-procedural locomotion becomes equal to t_{sp}^i, we start the motion-blending based locomotion at the corresponding time t_{mb}^i. We set the blending weights such that the blending algorithm starts with using $m_{mb} \in \mathcal{M}_{mb}$ to ensure smooth transition. Once the transition is complete, we set appropriate blending weights to achieve the desired velocity.

11 Comparison with Decoupled Systems

We evaluate our 2D navigation algorithm, MCN, with prior methods based on velocity obstacles (ORCA) and social forces (SF). In each case, we couple the 2D planner with a motion blending based synthesis algorithm and simulate at a fixed time-step. The MCN algorithm accounts for the full body pose during 2D planning leading to fewer collisions in the synthesized motion (Table 1 in the main document). Moreover, it generates smoother trajectories at a slightly higher run time cost. Provided here are details of the evaluation criterion and a discussion of the results.

11.1 Agent-Agent Collision Rate

We estimate the rate of agent-agent collisions by analyzing the 2D position of the bounding disc of the underlying articulated character. Our coupled approach ensures that the position of the disc is synchronized with that of the root joint of the corresponding skeletal mesh (Section 3.4 of the main document). Penetration Depth is a common metric used for quantifying collisions. It can be defined as the minimum displacement required to eliminate overlap between two entities. For each pair of adjacent time steps, we compute the maximum penetration depth (PD_{ij}) between two agents, i and j, over the interval bound by those time steps t^k and t^{k+1}. Assuming that agents move linearly between time steps, the position of agent i during the interval $[t^k, t^{k+1}]$ can be given as $\mathbf{p}_i(t) = \mathbf{p}_i + \mathbf{v}_i t, t^k \leq t \leq t^{k+1}$, where $\mathbf{p}_i$ and $\mathbf{v}_i$ denote the position and velocity respectively of agent i at time t = t^k. The maximum penetration depth can be computed by finding the minimum distance, or, equivalently, the minimum squared distance, between the two agents, i and j, over the time interval $[t^k, t^{k+1}]$ as:

$$d_{ij}^{min} = \min_{0 \leq s \leq 1} \|(\mathbf{p}_i + \mathbf{v}_i^{'} s) - (\mathbf{p}_j + \mathbf{v}_j^{'} s)\|^2, \tag{16}$$

where $\mathbf{v}_i^{'} = (t^{k+1} - t^k)\mathbf{v}_i$ and $s = -\frac{\mathbf{p}_{ij} \cdot \mathbf{v}_{ij}^{'}}{\|\mathbf{p}_{ij}\|^2}$ for relative position $\mathbf{p}_{ij} = \mathbf{p}_i - \mathbf{p}_j$ and relative velocity $\mathbf{v}_{ij} = \mathbf{v}_i - \mathbf{v}_j$. We can normalize the maximum penetration depth over the time step as $PD_{ij}^k = max(0, 1 - d_{ij}^{min}/r_{ij})$ where r_{ij} denotes the sum of the radii of the two discs. Finally, the collision rate C for the simulation can be computed by averaging all frames and agents:

$$C = \frac{1}{TN} \sum_{t=0}^{N} \sum_{i=0}^{N} \sum_{j=i+1}^{N} PD_{ij}^t, \tag{17}$$

where N is the number of agents, and T is the number of time steps.

Table 1 in the main document presents the collision rates of the three methods on different benchmarks (Section 7.2.1 of the main document). The MCN algorithm accounts for the current pose of the skeletal mesh and several human motion constraints to generate velocities that are amenable to motion synthesis. This two-way coupling reduces the mismatch between the 2D planning and full body synthesis, and reduces collisions. In case of the 2-way crossflow and 4-way crossflow scenes, MCN generates significantly fewer collisions compared to the social forces model, which is unstable in dense conditions at a time step suitable for interactive applications. MCN also generates fewer collisions than ORCA, which is stable and efficient but does not account for motion constraints of the underlying character. However, MCN leads to slightly higher collisions in bidirectional flow. This is likely because our synthesis algorithm does not support back-pedalling which causes agents to execute complete 180 degree turns if the new velocity is offset by more than 90 degrees from the current velocity. This behavior can be seen in bidirectional flow scene where two populations meet other head-on. ORCA and SF do not show as many collisions since agents abruptly alter their velocities to avoid collisions. This leads to noisy and unnatural motions, as depicted by the higher average acceleration values, but also reduces the collision rate. Ideally, agents should smoothly and effectively avoid collisions. We intend to address this in future work by introducing motions that support back-pedalling.

11.2 Trajectory Smoothness: Average Acceleration

Similar to the above section, we analyze the 2D trajectory of the root joint. Smaller accelerations are likely to generate smoother motions. The smoothness score A is simply the average acceleration over all the agents and all the simulation steps:

$$A = \frac{1}{TN} \sum_{t=0}^{N} \sum_{i=0}^{N} \|\dot{\mathbf{v}}_i^t\|, \tag{18}$$

where N is the number of agents, T is the number of time steps, and $\dot{\mathbf{v}}_i^t$ denotes the acceleration of agent i at step t.

In all benchmarks, MCN generates smoother trajectories compared to ORCA and SF, as indicated by the relatively low average acceleration scores listed in Table 1 in the main document.

11.3 Computation Time

We also compare the average 2D planning time for all three navigation methods. The 2D update time is marginally higher for MCN due to the additional constraints, as compared to the prior methods. Our algorithm can interactively compute collision-free paths for a few hundred agents, even in dense scenes such as the 4-way crossflow.