Hybrid Long-Range Collision Avoidance for Crowd Simulation

Abhinav Golas[1], Rahul Narain[2], Sean Curtis[1], and Ming C. Lin, *Fellow*, IEEE[1]

[1]University of North Carolina at Chapel Hill
[2]University of California, Berkeley

Abstract—Local collision avoidance algorithms in crowd simulation often ignore agents beyond a neighborhood of a certain size. This cutoff can result in sharp changes in trajectory when large groups of agents enter or exit these neighborhoods. In this work, we exploit the insight that exact collision avoidance is not necessary between agents at such large distances, and propose a novel algorithm for extending existing collision avoidance algorithms to perform approximate, long-range collision avoidance. Our formulation performs long-range collision avoidance for distant agent groups to efficiently compute trajectories that are smoother than those obtained with state-of-the-art techniques and at faster rates. Comparison to real-world data demonstrates that crowds simulated with our algorithm exhibit an improved speed sensitivity to density similar to human crowds.
Another issue often sidestepped in existing work is that discrete and continuum collision avoidance algorithms have different regions of applicability. For example, low-density crowds cannot be modeled as a continuum, while high-density crowds can be expensive to model using discrete methods. We formulate a hybrid technique for crowd simulation which can accurately and efficiently simulate crowds at any density with seamless transitions between continuum and discrete representations. Our approach blends results from continuum and discrete algorithms, based on local density and velocity variance. In addition to being robust across a variety of group scenarios, it is also highly efficient, running at interactive rates for thousands of agents on portable systems.

Index Terms—crowd simulation, collision avoidance, lookahead, hybrid algorithms

1 Introduction

Long-range vision is critical to human navigation; in addition to avoiding nearby obstacles, the human visual system looks ahead to perform dynamic global planning and local navigation. By considering the distribution of other pedestrians and obstacles over large distances, people can anticipate overcrowded regions and navigate around them, thereby finding efficient, uncongested paths to their goals. Thus long-range vision greatly improves crowd flow and progress. Most existing work addresses either global navigation around static obstacles or local avoidance of collisions with nearby pedestrians, but often neglects the importance of long-range collision avoidance. Modeling long-range collision avoidance holds tremendous potential, to improve the flow of simulated crowds and help them reach their goals faster. To maximize utility, such a model should improve crowd flow without disrupting existing crowd simulation pipelines. Thus we consider this as the primary goal of this work.

The state of the art for this topic is a synthetic-vision based steering algorithm proposed by Ondřej et al. [1]. This method explores a vision-based approach for collision avoidance among walkers. It offers global efficiency among the agents in terms of overall walking time. Achieving reasonable performance is perhaps the key challenge of using this approach for large-scale, interactive applications. Even a parallel, GPU-based implementation cannot handle more than 200 agents at interactive rates. Complementing this approach, our work addresses this problem by offering a simple and efficient alternative that naturally extends existing local collision avoidance algorithms to provide long-range collision avoidance. Our avoidance algorithm works based on the concept of *lookahead*, i.e. future agent states are approximated using past and present information; and these states are used to model possible collisions with agents not considered by local collision avoidance. Our method is robust even in presence of obstacles and chaotic crowd motion, and provides improved correspondence to real-world behavior.

Collision avoidance algorithms can be broadly classified into two categories: discrete and continuum – based on the underlying representation of crowds. We formulate and demonstrate our lookahead approach for both classes of algorithms, as the problem is not restricted to either class. Though our demonstration in this paper uses specific examples of continuum and discrete algorithms, our technique can be easily applied and generalized to other collision avoidance algorithms.

The use of continuum and discrete algorithms for collision avoidance also brings up a common issue with either class, namely their applicability to different ranges

golas@cs.unc.edu
narain@eecs.berkeley.edu
seanc@cs.unc.edu
lin@cs.unc.edu

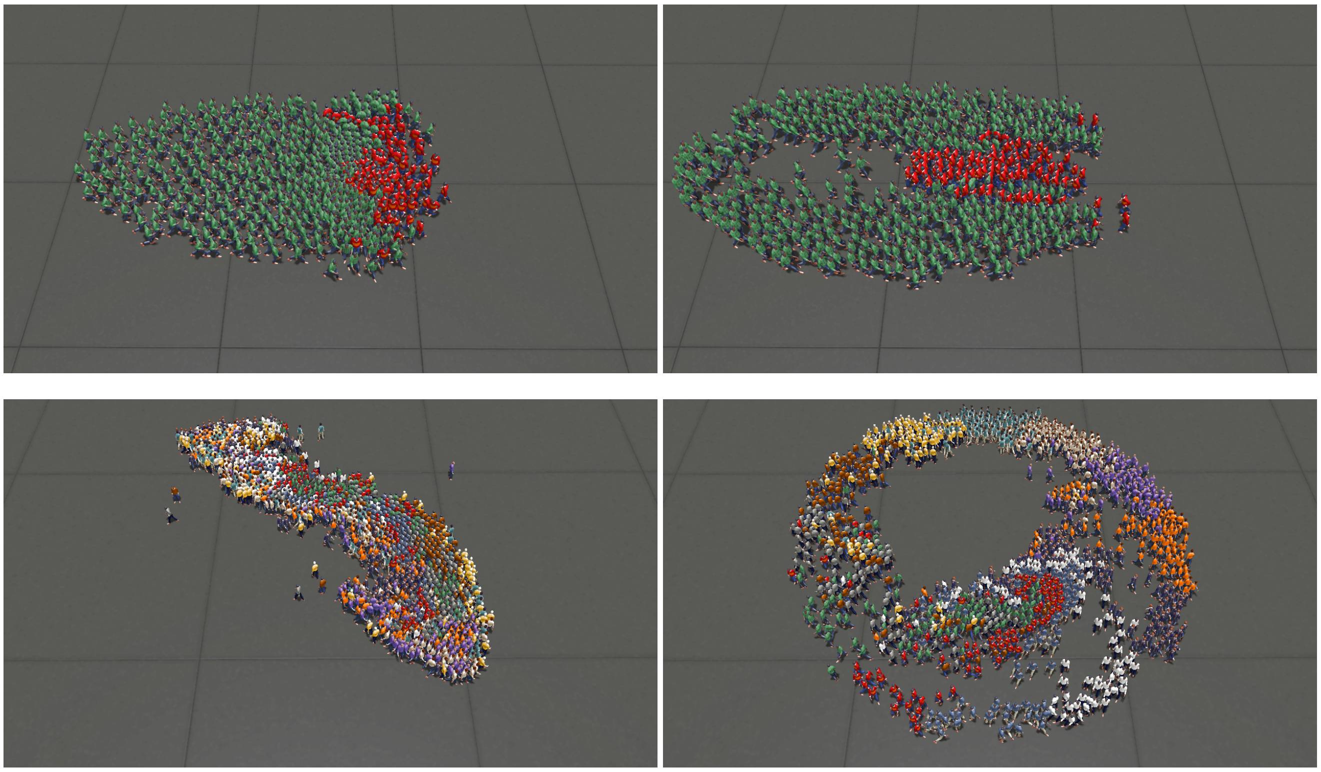


Fig. 1. Results without lookahead (left) and with lookahead (right) for 2 demo scenarios. **Crossing**: (Top) shows two groups of agents seeking to exchange positions at simulation time $t = 10\,s$. Note how, with lookahead, the bigger group parts to allow smaller group through. **Circle**: (Bottom) shows agents on the edge of a circle heading to diametrically opposite points at simulation time $t = 40\,s$. Note significantly improved progress with lookahead.

of agent density. Continuum algorithms (e.g. [2]) are ideally suited for medium to high densities, since the continuum assumption holds when pedestrian paths are tightly constrained by the nearby neighbors found at these levels of density. On the other hand, though discrete algorithms can be applied at any density, their computational costs escalate at high densities, along with numerical issues in some cases. Since crowds can exhibit an entire spectrum of densities even in any particular scene, these cases are not exceptions, but common occurrences. This insight suggests the need for an inexpensive hybrid scheme that locally blends both approaches for efficient collision avoidance over the entire spectrum of crowd densities. Such a scheme should choose the most suitable underlying algorithm for a particular simulation sub-domain, especially for problem cases where the wrong choice can lead to instabilities or other computational issues such as in scenarios of high or low densities, high variance of agent velocities, etc.

To address these problems, this paper introduces the following main results:

- A novel approach for approximate long-range collision avoidance that can be used with discrete or continuum algorithms with minimal increase in computational costs.
- An inconsistency metric to measure oscillations in agent trajectories that can be used to detect chaotic crowd behavior and curtail lookahead or to serve as a basis for comparing crowds, real or simulated.
- A hybrid algorithm that combines existing continuum and discrete collision avoidance algorithms to efficiently compute smooth local collision avoidance responses in any sub-domain.
- Comparison to real-world data that demonstrates improved speed sensitivity to density in simulated crowds using our algorithm, similar to human crowds as measured using the *fundamental diagram*.

Our results show significant improvements in crowd progress with minor increases in computational costs In Fig. 1, we demonstrate our approach on two scenarios, where improvements in crowd behavior and progress are seen with less than 3x computational overhead. Our approach is able to perform interactive long-range steering for both large, dense crowds and sparsely populated scenes, but also achieve interactive rates on a commodity laptop.

2 Background

We model a crowd as a set of agents, each of which has a specified goal position that it attempts to reach while avoiding collisions with other agents and with static obstacles in the environment. The standard crowd simulation loop that we and others often use is as follows:

1) For each agent, perform *global planning* to find a path to the goal that avoids collisions with static obstacles while ignoring other agents. Set the preferred velocity v_p along the direction of the initial segment of the path.
2) For each agent, perform *local collision avoidance* (LCA) to steer the preferred velocity v_p away from collisions with other agents, yielding the actual velocity v that the agent moves with.

Below, we briefly discuss some of the prior work relating to these two steps and discuss some of the data and techniques used for validating crowd simulation.

Most algorithms for global planning represent the connectivity of free space in the environment as a graph, and perform search queries for each agent to determine a collision-free path [3], [4], [5], [6], [7], [8], [9]. We do not diverge from previous work in this aspect.

A variety of models have been proposed for local collision avoidance among agents. These may use either discrete or continuum representations of the crowd. In discrete models, each agent considers other agents as individual obstacles, and attempts to avoid all of them simultaneously. Collision avoidance in this context can be formulated in terms of repulsion forces between agents [9], [10], [11], [12], [13], [14], [15], [16], or geometrically based algorithms [17], [18], [19], [20], [21]; the state of the art involves treating possible collisions as obstacles in velocity space [22], [23], [24], [25]. As considering the interaction of all pairs of agents becomes expensive in large crowds, such methods typically only take into account neighboring agents that lie within a specified radius, limiting the amount of lookahead possible. Guy et al. [26] propose a method to mitigate the computational cost of large neighborhoods by approximately clustering agents.

In a continuum-based approach, one first obtains from the set of agents a density field and a velocity field by accumulating the agents' positions and velocities on a background grid. This smoothed representation can then be used to compute the ideal motion of agents while avoiding regions of high density. The method of Treuille et al. [27] performs a global solve over the obtained density and velocity fields, giving compelling results including long-range congestion avoidance effects. However, its computational cost increases steeply with the number of distinct goals in the simulation, making this approach unsuitable for interactive crowd simulation where agents may have many diverse goals. Narain et al. [2] propose a technique that prevents overcrowding in highly dense crowds, but it relies on purely local information and thus cannot plan around congestion at a large distance.

Validating crowd simulation has always been challenging. Historically, the presence of so-called "emergent phenomena" has been considered evidence which suggested correctness. Steerbench is a suite of tests designed to allow comparison of models [28]. While it suggests some basis for comparing models, it does not present (or use) a ground truth; there is no data of human pedestrians used in performing the benchmarks.

In the pedestrian dynamics community, the most common quantitative metrics for crowd behavior deal with aggregate crowd properties: flow and density. The relationship between flow and density has been referred to as the "fundamental diagram" [29]. In addition to this aggregate analysis, Guy et al. propose a new statistical metric for measuring how likely a particular pedestrian model is to match a given set of data [30]. In recent years, experiments have been performed with human subjects in various scenarios and several data sets have been made publicly available: "one-dimensional" pedestrian movement, uni-directional movement [31], uni- and bi-directional flow in a corridor [32], [33], and flow through a bottleneck [34]. The value of lookahead is greatest in the case of conflict. As such, we do not perform validation against the uni-directional corridor or bottleneck flow. Instead, we perform analysis on an experimental setup similar to [33]. Ideally, we would prefer data of pedestrians moving over a large space, however, even in this limited scenario, we can show that lookahead improves the behavior of the simulated crowds (see Fig. 12).

Our approach aims to extend some of the existing work in LCA algorithms to support long-range collision avoidance queries. We accomplish this through the simple yet effective approach of extrapolating agents' motion into the future. Our algorithm is described in section 3, and we demonstrate its application to continuum and discrete algorithms in section 3.1 and section 3.2. In some cases lookahead may not be possible, particularly in presence of obstacles and turbulent flow. These are detailed in section 4 in addition to a novel metric for measuring oscillation and chaotic behavior in crowds. Furthermore, using discrete models alone can be extremely expensive in dense crowds, while continuum models are poorly suited to representing the motion of sparse crowds. In section 5, we propose a hybrid algorithm that blends results from continuum and discrete algorithms, producing consistently realistic results for both low and high densities under various velocity conditions. We demonstrate the advantages of our proposed techniques with examples in section 6, and compare our proposed lookahead based long-range collision avoidance algorithm with real-world data in section 7. Finally, we conclude with the limitations of our method, and discuss avenues for future work in section 8.

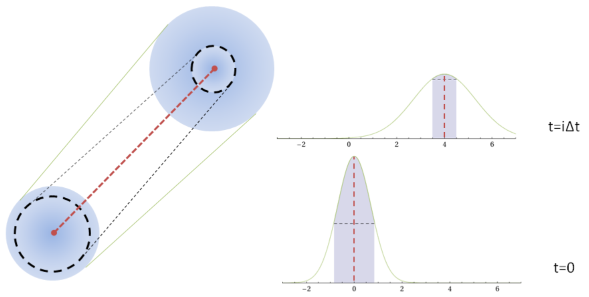


Fig. 2. Effect of extrapolation in time from $x = 0, t = 0$ to $x = 4, t = i\Delta t$. Dotted line indicates effective radius ($P \geq 0.4$) of agent for collision avoidance, while spread of gaussian curve indicates splatting area for density field creation.

3 Lookahead for Long-Range Collision Avoidance

In this section, we describe our approach for efficiently computing long-range collision avoidance for both continuum-based and discrete agent-based crowd models. The problem can be summarized as follows: For each agent with a given preferred velocity v_p (as computed by the global planning stage), we wish to find an updated velocity v close to the preferred velocity v_p that avoids congestion in front of the agent at a range of distances from far to near, and also avoids collisions with neighboring agents. Influences from nearer agents should receive priority; that is, the agent should not divert itself to collide with a nearby agent in trying to avoid congestion farther away.

Given the extensive amount of already existing work on *local* collision avoidance algorithms, our aim is to take advantage of these existing techniques to solve the problem of *long-range* collision avoidance. In this paper, we propose a general approach for decomposing long-range collision avoidance into a sequence of simple LCA queries. Thus, our algorithm can re-use existing LCA algorithms with minimal increase in computation and coding effort. We show how to apply this approach to both the discrete and continuum settings, resulting in efficient algorithms for long-range collision avoidance in both cases. The crux of the idea lies in evaluating LCA queries not only on the current state of the crowd, but on its future state, estimated at a series of future times, enabling greater lookahead while using only local operations.

When an agent plans its long-term motion, it needs to estimate the motion of other agents over a large time interval into the future. While the future motions of other agents are of course unknown, they can be estimated with some degree of confidence using the agents' current velocities. To reflect the uncertainty in this estimation, we treat an agent's predicted location in the future not directly as a point, but as a probability distribution representing the expected probability of finding the agent at a given position. Intuitively, one can think of this as a traveling "blob" of probability, whose center $x(t)$ is linearly extrapolated from the agent's current position and velocity, and whose spread $\sigma(t)$ gradually expands over time, reflecting the increasing uncertainty as one looks further in the future.

In the continuum representation of the crowd, this has the effect of smoothing out the influence of any agent on the crowd density field, which enlarges the distance over which it influences the motion of other agents while simultaneously attenuating the magnitude of its effect. Thus, when an agent performs a short lookahead, only its nearby agents are significantly influential, while over a large lookahead, it only sees a smoothed-out density field that averages over many agents across a large area. In the discrete model, an agent is treated as a rigid, impenetrable "blocker" of fixed radius. When the agent position is uncertain, we consider a point to be blocked by the agent if the probability that the agent covers that point is at least a certain threshold p. As can be seen in Fig. 2, as the uncertainty increases, the effective size of the blocker decreases. This has the desirable effect that agents planning far into the future are not excessively hindered by the estimated motion of other agents, given that the latter is unreliable.

With this model for uncertainty, we can formulate the basic lookahead algorithm for long-range collision avoidance. The algorithm starts with the preferred velocities v_p obtained from the global planning stage, and performs a number of iterations $i = i_{\max}, i_{\max-1}, \ldots, 0$ with decreasing time horizons $\Delta t_i = 2^{i-1}\Delta t, i > 0$, and $\Delta t_0 = 0$. In each iteration, we extrapolate the state of the crowd by a time interval Δt_i into the future, perform an LCA query (with uncertainty) using the preferred velocity, and then replace the preferred velocity with the result of the LCA, as illustrated in Fig 3. In the last iteration, we set the lookahead Δt_0 to zero, so that we perform the standard LCA with no uncertainty, and thus maintain the collision avoidance guarantees of the underlying LCA.

With this scheme, agents are sampled in a larger radius than in the standard LCA query, and extrapolated queries are biased towards the direction of motion, providing lookahead. Our approach smoothly merges the effects of distant and nearby avoidance considerations. Congestion avoidance with a long time horizon takes place in earlier levels, influencing the final result by updating the preferred velocity; nevertheless, this can still be overridden if needed to avoid imminent collisions with nearby agents, which are considered later in the process.

The algorithm is defined formally in Fig. 4, where we denote by $v = A(v_p, v_c, x, \rho)$ an LCA query performed for an agent at position x with current velocity v_c and preferred velocity v_p in a region of local density ρ (people per unit area), producing a collision-free velocity v. In the following subsections, we apply

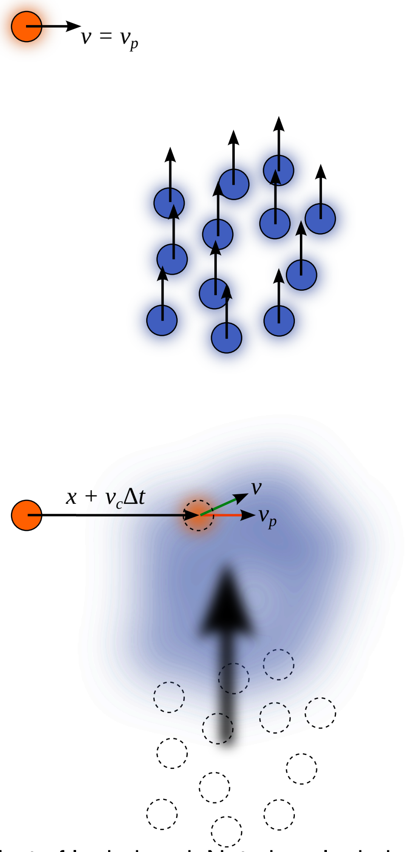


Fig. 3. Effect of lookahead. Note how lookahead allows the orange agent to see the approaching crowd and adjust its velocity from preferred velocity v_p to v by incorporating information from the future crowd state at time $t + \Delta t$.

```
For each leaf node $p$
  • Foreach level $i$ in range $i_{\max}$ to 0 DO
      1) Determine future state of crowd $x_i = x + v_c\Delta t_i$
      2) Solve local collision avoidance problem $v = A(v_p, v_c, x_i, \rho_i)$
      3) Update preferred velocity $v_p = v$
  • END Foreach
```

Fig. 4. Lookahead Algorithm using LCA algorithm A.

our generic long-range collision avoidance algorithm to two examples of LCA algorithms, one continuum-based and one discrete, showing the broad applicability of our technique. We also describe some optimizations that are possible in the specific cases.

3.1 Continuum Lookahead

Continuum representations treat the crowd as a continuous distribution of density ρ and velocity v over space, through which any given agent must navigate. The knowledge of the density distribution enables us to determine congestion directly as regions of high density. It is well known that pedestrians walk slower in regions of high density [35], a fact that can be formalized into a graph known as the *fundamental diagram* relating density, ρ, to a natural walking speed: $v_{\max} = f(\rho)$ Therefore, agents should navigate around overcrowded regions to avoid lowering their speed and maximize their rate of progress towards their goals. In this section, we first describe a simple algorithm that uses this idea to avoid congestion over a short time horizon, then extend it to look much further in time using our long-range approach.

Consider an agent that has a preferred velocity v_p pointing towards of the goal. Suppose over the planning time horizon Δt, the agent maintains a constant heading along a chosen direction $\hat{v}$ and walks at the maximum speed allowed by the fundamental diagram f. Then to first order, the density it passes through will change at a rate of $f\hat{v} \cdot \nabla\rho$, and so its displacement after time Δt will be

$$d(\Delta t) = f\hat{v}\Delta t + \frac{1}{2}(f\hat{v} \cdot \nabla\rho)f'\hat{v}\Delta t^2, \tag{1}$$

where f and f' are evaluated at the density at the current position. We choose $\hat{v}$ to maximize the progress towards the goal, $v_p \cdot d(\Delta t)$. This formulation reduces to the following optimization problem,

$$arg\max_{\hat{v}} \left(v_p \cdot \hat{v} + \frac{f'\Delta t}{2}(\hat{v} \cdot \nabla\rho)(v_p \cdot \hat{v}) \right) \quad \|\hat{v}\| \leq 1. \tag{2}$$

We can solve this problem using projected gradient descent, with the direction of the current velocity as the initial guess; this converges in less than ten iterations on average. This simple approach produces excellent avoidance results with maximal progress while still being computationally inexpensive. Though similar in spirit to [2], it avoids the need to calculate a global pressure to exert forces.

Before formulating the lookahead algorithm for continuum crowds, we first need to estimate the future densities of the crowd. In accordance with the uncertainty model, extrapolation further into the future requires that each agent's contribution to the density field be spread out over larger and larger areas, which can become inefficient with the traditional "splatting" approach. Instead, it is more efficient to represent future states on coarser grids, which will automatically have the effect of increasing the agents' effective footprint. Each successive grid is coarsened by a user-defined factor c, which represents the increase in uncertainty $\sigma(\Delta t)$ from one level of lookahead to the next. Thus, a pyramid of grids is constructed, where each level is coarser than the one below it by a factor c. Level i of the pyramid contains the future state of the crowd at time $t + \Delta t_i$.

With this representation, the lookahead algorithm as defined in Fig. 4 can be directly applied. For each cell at the bottom of the pyramid, we solve the LCA problem separately at multiple levels of the pyramid, starting from the top and cascading the solution at level i as the preferred velocity at level $i - 1$. Though this involves redundant computation being a depth-first approach, we prefer this approach as opposed to the breadth-first model where all cells at level i are solved, and the solution is cascaded down to level $i-1$. This allows improved parallelism since there is no interdependence of solutions of neighboring cells, a reduced memory footprint since intermediate solutions do not need to be stored, and reduced interpolation and smoothing artifacts.

3.2 Discrete Lookahead

We now extend the algorithm to discrete collision avoidance. Here, we use an extended reciprocal velocity obstacle (RVO) algorithm [25], which is implemented in the RVO2 library [36]. The RVO algorithm performs collision avoidance in velocity space, that is, the space of possible velocities that an agent may choose. In this space, we create a "velocity obstacle" for each neighboring agent, which represents the set of velocities that would lead to a collision with that agent. Then, choosing a velocity outside the union of all these obstacles ensures collision avoidance. Each obstacle is modeled as a constraint in a linear optimization problem to determine a collision-free velocity closest to the preferred velocity.

The RVO library requires the choice of a neighborhood radius R and time horizon τ, and only guarantees collision-free behavior within time τ with nearby agents no further than R distance away. This technique is limited to local planning in a small neighborhood, as increasing τ and R to large values degrades the performance of the method. To support long-range collision avoidance, we apply our algorithm to RVO-based LCA with minor modifications.

Instead of constructing trees for each future instant, we approximate future neighbor searches from the current state. Recall that our generic algorithm has multiple levels, and at the ith level, we consider a lookahead of Δt_i time into the future. In the discrete setting, we search for agents which may collide with the current agent within time Δt_i. Since distance between two agents can change at most by $2v_{\max}\Delta t_i$ in this time, where $v_{\max}$ is the maximum agent speed, the agents relevant at level i are those that lie at distances between $R + 2v_{\max}\Delta t_{i-1}$ and $R + 2v_{\max}\Delta t_i$ from the current agent at the present time. Once these neighbors are determined, we create velocity constraints using the agents' extrapolated future positions, with their effective radii r_f reduced by a ratio c for every step into the future. (This constant is the same as that grid coarsening factor for the continuum formulation, since both density and effective agent radius are inversely proportional to the standard deviation of the probability distribution assumed for the agent.)

Now, all the long-range interactions considered at different levels are represented simply as constraints on the final velocity of the agent. Instead of solving

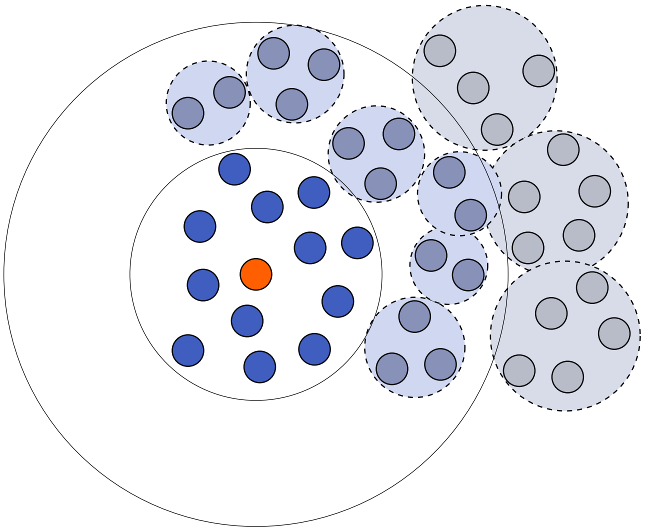


Fig. 5. Distant agents can be clustered for collision avoidance, cluster size being proportional to distance. Since possible collisions with distant agents lie in future timesteps, extrapolated future agent states have high uncertainty, and hence small effective radii, making individual avoidance inefficient.

the levels one after another, we may apply all the constraints simultaneously in a single RVO optimization. This means that only one optimization solve needs to be performed per agent, but at the expense of an increase in the number of constraints. We therefore adopt a level-of-detail approach to reduce the number of constraints by adaptively grouping distant agents into clusters.

As we extrapolate agents further in time, their effective radius reduces further, and thus have a decreased effect on agent velocity. Thus, it is prudent to cluster these agents both to improve efficiency, and to increase the probability of avoiding a future collision. We use a spatial hierarchy, such as a $k-d$ tree, over the agent positions to choose the clusters. Such a hierarchy already exists in the RVO implementation to support nearest neighbor queries, and so does not require additional computational effort. The nodes of the tree can provide suitable candidates for agent clusters.

When considering lookahead at level i, that is, until future time Δt_i, we only consider agents at a distance between $R + 2v_{\max}\Delta t_{i-1}$ and $R + 2v_{\max}\Delta t_i$. These agents should be grouped into clusters of size $\frac{\Delta t_i}{\Delta t}$ as shown in Fig. 5. Instead of performing multiple searches to collect nodes at each level, we perform one tree traversal where the level of the node can be determined based on its distance from the agent. Thus, we perform a tree traversal where at any node C, we can determine its level i by checking which distance band it lies in, i.e. $d_C \in [R + 2v_{\max}\Delta t_{i-1}, R + 2v_{\max}\Delta t_i]$. However, every node may not form a good candidate since the distribution of agents in the subtree of this node may be sparse. Therefore we use a maximal separation as a quality measure

```
For each agent $j$, traverse the tree $T$ starting from the root node, at each node $C$:
  • If node $C$ does not satisfy maximal separation constraint recurse on its children
  • If constraint is satisfied and its level $i \leq i_{max}$, formulate velocity obstacle constraint for node $C$
where $i_{max}$ is the highest tree level considered.
```

Fig. 6. Lookahead Algorithm using RVO.

of each node, i.e. $\max_{i \in subtree(C)} dist_i$, where $dist_i = \min_{k \in subtree(C)}(\|x_i - x_k\|)$. Though exact computation of this value is expensive, we compute an approximate value during tree construction by choosing the maximum of child values, and the separation between the nodes themselves. Thus, if a node satisfies this quality constraint, it can be added as a velocity constraint.

Once we have a chosen a set of agents to form a cluster, we set its position x_C and velocity v_C as the mean of the positions and current velocities of its member agents. We choose the effective radius r_C of the cluster so that it covers all the expected agent positions, and is padded by the effective agent radius r_f for time $t + \Delta t_i$. In other words, for a cluster of m agents $\{x_1, x_2, \ldots, x_m\}$, we define

$$x_C = \frac{1}{m} \sum_{j=1}^{m} x_j, \tag{3}$$

$$v_C = \frac{1}{m} \sum_{j=1}^{m} v_j, \tag{4}$$

$$r_C = r_f + \max_j \|x_j - x_C\|. \tag{5}$$

With this definition, we define our discrete lookahead algorithm in Fig 6.

4 Curtailing Lookahead

The discussion thus far has been based on the assumption that agent states can be extrapolated to any future time. However in certain cases, the information available to long-range collision avoidance may be insufficient to create a reliable future state. In such cases, we curtail lookahead to the last reliable future time.

If the extrapolated path intersects an obstacle, we can infer that the agent has a plan to prevent this. However, determining this plan would require a query to the local planner, which in turn would require positions of all neighbors. This can recursively trigger local planning queries for an increasing number of agents, thus increasing computational cost. Thus choosing the simpler alternative of curtailing lookahead is a computationally efficient choice. This also has parallels among human crowds, which tend to plan within their visual range. Since obstacles restrict the visual range, it is natural to allow obstacles to curtail the planning region in a simulation as well.

The other scenario where we propose curtailing lookahead is when an agent has a chaotic trajectory - as measured over a small window of previous time steps. Extrapolating such a trajectory using any low-order polynomial function is likely to result in large approximation errors. This may even be indicative of a chaotic crowd, or an artifact in the underlying algorithm. In either case, the local crowd state is not amenable for long-range collision avoidance, as its future state is hard to predict.

We now detail the exact method utilized to address these two cases.

4.1 Obstacles

We allow extrapolation step i if and only if the path from the agent's current position to its extrapolated position does not intersect any obstacles, which can be modeled as a visibility query. Correct extrapolation of the agent's position would necessitate a model for agent response to an obstacle. As previously explained, this can result in a significant increase in computation. In addition, this may require information from the global planner. This information is typically not available. While it could be made available, this would require significant changes to the simulation pipeline, undercutting one of the goals of this approach: to extend current systems with minimum modification.

These queries can be efficiently implemented for both continuum and discrete simulations. Continuum simulations traditionally model obstacles as distance fields. These can be used to efficiently perform the needed intersection tests. The extrapolation curve representing the expected trajectory can be checked for collisions by finding the minimum distance to any obstacle in the scene. Such algorithms involve sampling the distance field at multiple points - either uniformly or adaptively - along the curve, with sampling controlled by the cell width of the distance field and the curvature of the curve itself. This approach can also be used for discrete simulations. However, in case the underlying simulation represents obstacles as discrete objects, an alternate approach can be used. In such a scenario, the problem can be modeled as a ray shooting problem. Using a hierarchical structure for static obstacles in the scene, such visibility queries can be performed efficiently at run-time.

4.2 Chaotic Crowds

Extrapolation assumes that agent velocity is temporally consistent, i.e. the agent is expected to follow a predictable path. However, if the underlying crowd flow presents chaotic disturbances, then future agent states cannot be determined reliably with low-order extrapolation. Determining whether a crowd exhibits chaotic behavior locally or globally requires the knowledge of agent velocities for a time window. Given this information, we propose a new 'inconsistency metric' to detect chaotic behavior, and measure the suitability of the agent's state to extrapolation.

4.2.1 *Inconsistency Metric*

In case of an oscillating agent trajectory, extrapolation of an agent's position is not feasible due to the uncertainty in the agent's velocity. This oscillation can be measured by the acceleration of the agent, determined using its first-order approximation as

$$\delta v = \frac{v^i - v^{i-1}}{\Delta t}, \tag{6}$$

where v^i is the agent velocity at step i, and Δt is the time step between the two steps. Thus for each agent, a relative deviation can be computed at every time step i.

The inconsistency metric is computed using a set of n deviation vectors. These 2-dimensional deviation vectors are concatenated into a $n \times 2$ matrix M. We use Principal Component Analysis (PCA) to analyze the deviation vectors by computing the eigenvalues of the matrix:

$$C = \frac{1}{n} M^T M. \tag{7}$$

The eigen-decomposition of this matrix is of the form:

$$X^T \Lambda^2 X, \tag{8}$$

where Λ is a diagonal matrix with entries λ_1 and λ_2, as shown in Fig. 7. The metric is then defined as the sum of the magnitudes of the eigenvalues:

$$\sigma = |\lambda_1| + |\lambda_2| \tag{9}$$

Low values of the metric indicate a smooth low-order trajectory of the agent, while higher values indicate chaotic paths. The rows of X give the corresponding eigenvectors x_1 and x_2, which can be used to further analyze the nature of the deviation.

To determine whether an agent can be accurately extrapolated, the inconsistency metric can be computed on a local neighborhood of the agent. In this case the matrix M is constructed from the deviation vectors of the agent and its neighbors over a time window of τ seconds, i.e. for k agents, data for f time steps, $f = \frac{\tau}{\Delta t}$ is concatenated into a $kf \times 2$ matrix M. The contribution of older samples can be also weighted to facilitate graceful degradation. For analysis of the entire crowd, the metric can be computed for all agents using a single matrix containing all deviation vectors for the given time window. Such analysis can be useful for comparing collision avoidance algorithms, preferring those which result in smoother paths as indicated by slowly varying velocity.

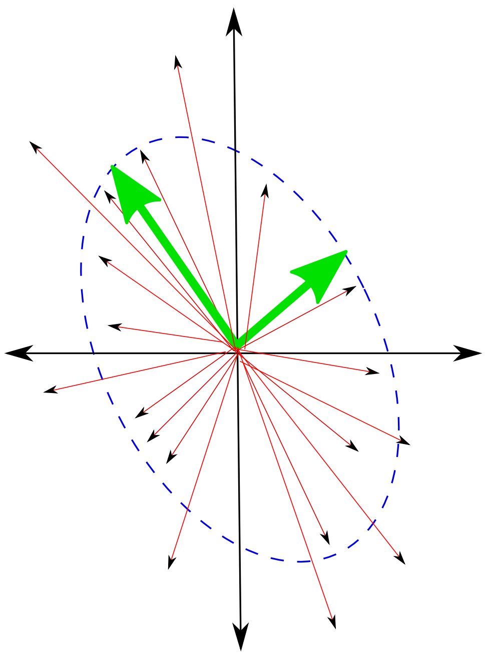


Fig. 7. Our proposed inconsistency metric σ is computed as the sum of the eigenvalues λ_1, λ_2 of the given deviation vectors δv (in red), which represents the variance of these deviations. The eigenvectors x_1, x_2 (in green) corresponding to these eigenvalues represent the principal components of the space of deviations.

4.2.2 *Curtailing*

The inconsistency metric value as computed for the local neighborhood of an agent can be used to curtail the extent of lookahead. Variation of velocity in the vicinity of the agent indicates that the agent velocity is likely to change in the near future as well. For inconsistency values near zero, the agent can perform the allowed maximum $i_{\max}$ lookahead steps. As the metric value rises to a maximum $\sigma_{\max}$ and beyond, the maximum allowed lookahead steps decrease to zero, at which point only local collision avoidance is performed. We use linearly spaced transition points, performing at most i lookahead steps if the metric value σ lies in the range:

$$\sigma \in \left[\frac{(i-1)\sigma_{\max}}{i_{\max}} \quad \frac{i\sigma_{\max}}{i_{\max}} \right] \tag{10}$$

for simplicity. The agent's choice of how far to look ahead is assumed to be based on local crowd conditions. Thus, only neighbors as defined by the local collision avoidance algorithm are used for computing the inconsistency metric for an agent. This metric value controls the agent's lookahead steps, as well as the lookahead of any agents considering this agent for collision avoidance. This implies that if a distant agent cluster has a metric value higher than that allowed for the lookahead step, then that cluster will not be considered for collision avoidance at that step, since that cluster cannot be expected to behave coherently as a single entity. Also, though the metric can be computed for every agent, its value is expected to vary smoothly, hence it can be sampled at certain points and interpolated for remaining agents, improving the efficiency of this computation.

5 Hybrid Crowd Simulation

Collision avoidance guarantees, provided by algorithms discussed thus far, are conditional on certain assumptions. In situations where these assumptions are violated, collision avoidance guarantees do not hold, and this can produce incorrect or at least visually unappealing results. For example, continuum algorithms work on the assumption that a crowd can be represented accurately as a density field. In low density regions, where this assumption does not hold, agents routinely collide with each other, or have to be pushed apart creating oscillatory behavior. In addition, grid representations of these fields can suffer from aliasing issues, resulting in damped or smoothed velocities. Discrete algorithms suffer from numerical issues at high densities, due to low inter-agent separation. For example, force based methods use repulsive forces that are inversely proportional to distance. As a result, strict limits need to be enforced to prevent agents from colliding, either on the time step, or on the repulsive forces themselves. In geometric methods like RVO [22], [25], this is reflected in increased constraints on the solution, meaning that the algorithm needs to spend more computational time to converge, or risk failing to compute a collision-free velocity. In addition, computational costs of discrete algorithms are proportional to the number of agent neighbors. Since this cost rises sharply for high density regions, discrete algorithms can lose their performance edge for such scenarios.

Our lookahead formulation performs successive local collision avoidance queries, thus such errors are likely to accumulate and cause significant issues. To address this issue, we propose a simple and efficient hybrid algorithm that blends discrete and continuum collision avoidance results. This is possible since problem cases for either class of algorithms do not overlap. The choice of algorithm is based on both density and variance in velocity. For varying density, there are three possible cases:

- $[0, \rho_{cmin}]$: Discrete collision avoidance
- $[\rho_{cmin}, \rho_{dmax}]$: Blend Discrete and Continuum collision avoidance
- $[\rho_{dmax}, \rho_{max}]$: Continuum collision avoidance

where ρ_{dmax} is the maximum density at which discrete collision avoidance can be applied, ρ_{cmin} is the minimum density for continuum collision avoidance, and $\rho_{cmin} \leq \rho_{dmax}$. By using linear blending, this can be expressed as:

$$v = v_{disc}(1-w) + v_{cont}w \tag{11}$$

$$w = w_\rho = clamp\left(\frac{\rho - \rho_{cmin}}{\rho_{dmax} - \rho_{cmin}}, 0, 1\right) \tag{12}$$

where $clamp(x, min, max)$ clamps the value of x to the range $[min, max]$, and v_{disc} and v_{cont} are collision-free velocities generated by discrete and continuum algorithms respectively. In our examples we use $\rho_{cmin} = 2$, $\rho_{dmax} = 4, \rho_{max} = 5.5$. The values of these parameters are guided by the densities observed in human crowds, being less than 5.5 people per m^2 in most cases. For choosing ρ_{cmin}, we rely on guidance from [37], which states that the crowds can be represented as a continuum "*provided the characteristic distance scale between pedestrians is much less than the characteristic distance scale of the region in which the pedestrians move*". In medium-scale examples like those shown in this paper, densities above $\rho_{cmin} = 2$ people per m^2 satisfy this constraint in the average sense. At higher density ranges, we observe qualitatively similar results using continuum and hybrid algorithms at densities higher than 4 people per m^2, which guides the choice of ρ_{dmax}.

To address the case of high velocity variance, we can define similar linear blending weights. In this case, blending weights are controlled by the standard deviation σ_v of the local velocity. We blend discrete and continuum velocities in a user-defined range $[c_1 v, c_2 v]$, where v is the local velocity, and c_1, c_2 are constants. In regions of low velocity variance, continuum avoidance is preferred, with discrete avoidance preferred in regions of low variance, where variance is σ_v^2. Then, in a manner similar to equation (12), we can define blending weights for this case as well:

$$w_{\sigma_v} = clamp\left(\frac{c_2 v - \sigma_v}{(c_2 - c_1)\, v}, 0, 1\right) \tag{13}$$

Note that this weight has a slightly different form to maintain the convention in (11).

We now need to combine these two weights to produce a single interpolation weight. Though a number of possible combinations exist, we choose $w = w_\rho w_{\sigma_v}$. This is biased towards a discrete solution, which ensures that the likelihood of regions with high velocity variance being simulated with continuum methods remains low. This weighing can be tuned by appropriately choosing c_1 and c_2. We find best results by choosing $c_1 = 1$, $c_2 = 2$. It is important to note that these two weights are not independent. As has been noted by [2] and others, velocity variance decreases at high densities. Thus adding a weight for velocity variance does not affect high density behavior significantly.

6 Results

Our algorithms were implemented in C/C++ using scalar code. In Table 1, we provide running times on a quad-core Intel Core i7 965 at 3.2GHz. Note that these times can be significantly improved by using appropriate vector instructions. We modified the RVO2 library to remove the restriction of maximum neighbors so

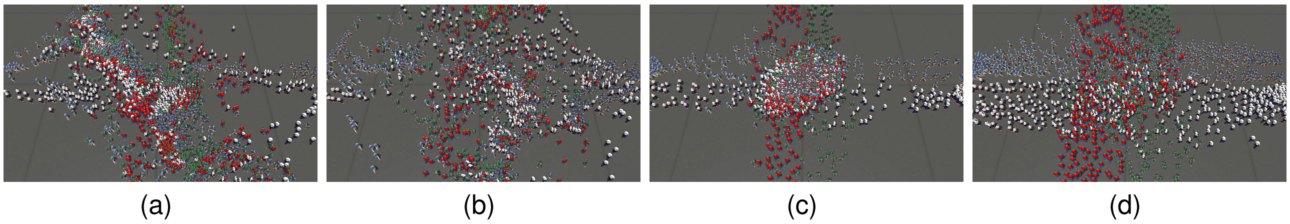


Fig. 8. **4 way** crossing of agents with 2000 agents. (a) Discrete (b) Discrete with lookahead (c) Continuum (d) Continuum with lookahead. Note lack of agent buildup in cases with lookahead.

Scene	**#Agents**	**Simulation**			**Time per Step (ms)**
		Method	**Duration (s)**	**Improv -ement**	
Crossing	500	Disc	135.2	2.6x	0.98
		Disc LA	52.5		3.68
Circle	1000	Disc	347.2	2.2x	2.6
		Hyb LA	156.0		5.2
4 way	2000	Disc	-	-	5
		Disc LA			47.3
		Cont LA			6.8
		Hyb LA			16.29
4 groups	2000	Cont	107.1	1.5x	6.6
		Cont LA	72.28		6.89

TABLE 1
Single thread performance for our examples ($dt = 0.01s$). **Legend**: **LA** - with lookahead, **Disc** - Discrete, **Cont** - Continuum, **Hyb** - Hybrid. **Note**: > 20 fps performance in all cases (> 60 fps w Hyb) and 1.5x - 2.6x reduction in simulation duration with LA

as to provide collision avoidance guarantees for the neighborhood threshold supplied.

Our lookahead and hybrid algorithms were tested on a number of cases. The first example scenes demonstrate the lookahead algorithm for the continuum and discrete cases. In the discrete case, two groups of agents – one bigger than the other – head towards each other on a collision course. Using our lookahead algorithm, the larger group of agents parts to allow the smaller group to go through, which is not observed in the traditional RVO algorithm. Though we encounter a 3.5x slowdown, the progress seen by agents is more than double, thus over the time of the simulation, the overall cost is less than 2x. In the continuum case 10, where 4 groups of agents attempt to reach diametrically opposite regions, more distant agents avoid the high density region in the center. In contrast to a simulation without lookahead, agents are able to reach their destination sooner, demonstrating improved crowd flow and progress. An advantage of the continuum case is that lookahead is extremely inexpensive, as is seen with this example, where significant improvements in behavior can be seen at almost no cost

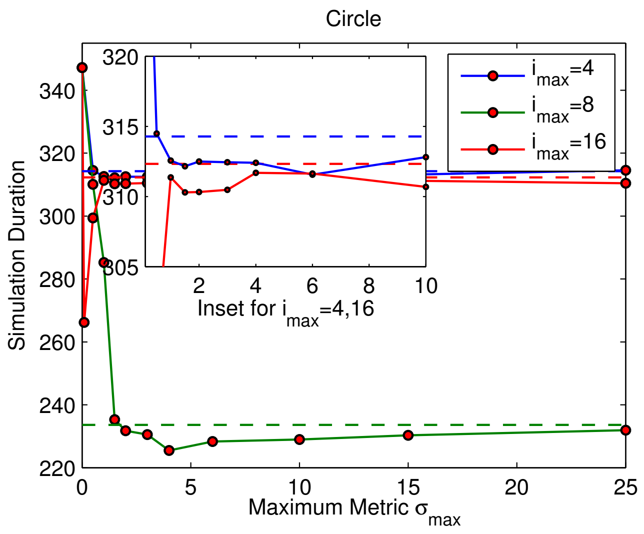
The second example scene demonstrates a 4-way crossing. With traditional collision avoidance schemes, a bottleneck quickly develops in the middle of the scene hindering progress and causes spurious behavior. Such behavior is significantly reduced with lookahead, both in the discrete and continuum cases. Using hybrid algorithm in this case provides two benefits. Oscillation of agents in low density regions is reduced as compared to the continuum algorithm, while significant performance benefits are obtained vs. the discrete algorithm. Though discrete lookahead slows down significantly due to the number of neighbors to be considered, the hybrid algorithm shows a performance benefit of 3x while retaining the same behavior. A visual comparison in Fig. 8 shows the difference in the 4 kinds of simulation.

We replicated the well-known circle demo with 2000 agents. In this scenario, agents are seeded on a circle and attempt to reach the diamterically opposite point as shown in Fig. 1. At a 2x extra computational cost, we observe significant improvement in behavior and progress. Agents are able to reach their goal in less than half the time, which balances the computational cost.

To quantify the benefit of curtailing lookahead we analyze the circle (Fig. 1) and 4-groups (Fig. 10) demos. We examine how the choice of $\sigma_{\max}$ and $i_{\max}$ affects the simulation duration, i.e. the total time taken by all agents to reach their goals. Fig. 9 shows the results of this experiment. In each graph, dotted lines represent simulation duration of constant lookahead, while solid lines represent curtailed lookahead. In both scenarios, as $\sigma_{\max}$ increases, overall performance improves beyond constant lookahead, as demonstrated by lower simulation durations. As $\sigma_{\max}$ continues to increase, simulation performance asymptotically approaches constant lookahead. It is clear that constant lookahead in scenarios with inconsistent behavior causes agents to become overly conservative, reducing the efficiency of crowd flow. By curtailing lookahead with appropriate parameters, agents reach their goals more efficiently, providing improvements of $10-100\%$. In addition to improvements in simulation performance, curtailing lookahead also results in reduced computation.

Evaluation of the inconsistency metric for curtailing lookahead increases computation by less than 5%, and this extra computation is balanced by the reduction in number of lookahead steps. Performance can be improved further by coarser evaluation, as noted in Section 4.2.2.

Most demos in this video use a maximum of 8 lookahead steps. As demonstrated by Fig. 9, differ-



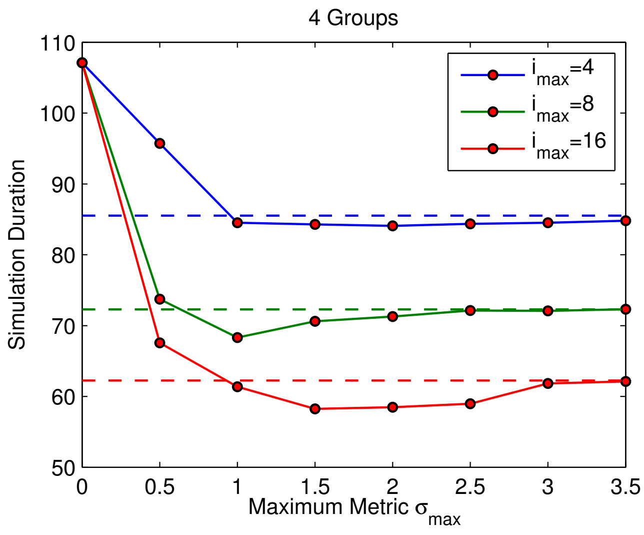
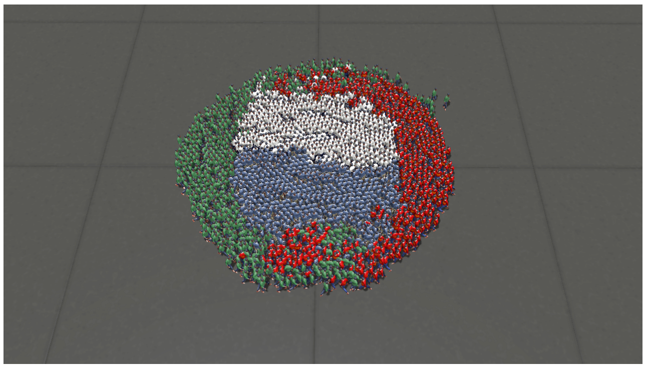
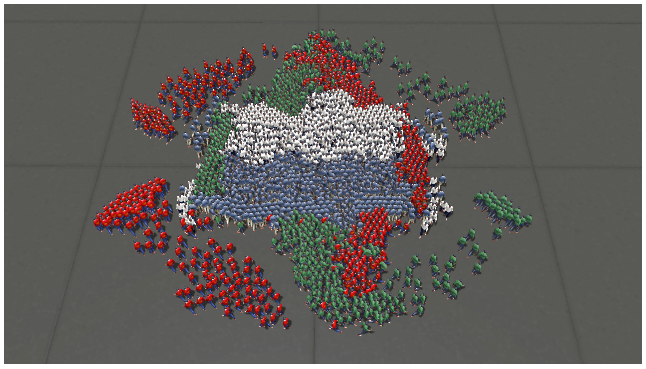


Fig. 9. Effect of Curtailing Lookahead in the circle (Top) and 4 groups (Bottom) demo. Solid lines show curtailed lookahead, while dotted lines show constant lookahead. Note how curtailing lookahead with a maximum metric value $\sigma_{\max}$ allows agents to reach their goals faster, as demonstrated by the reduced duration. In some cases, curtailing can double the improvement shown by lookahead (circle, $i_{\max} = 16$). Even with optimal choice of $m_{\max}$, we see benefits of 10%.



(a)



(b)

Fig. 10. **4 groups** of agents in circular formation exchange their positions. Notice how lookahead (b) shows red and green agents moving around the built up region in the center and avoid getting stuck as is the case in (a).

ent scenarios and collision avoidance algorithms can demonstrate best performance for different values of $i_{\max}$ and $\sigma_{\max}$. Our experiments indicate that $i_{\max}$ and $\sigma_{\max}$ can be optimized one at a time in order. This task if simplified by the fact that these optimizations do not exhibit local minima.

7 Comparison to Real-World Data

We compare the simulation result of our proposed algorithm against available open-source data. We use the data provided by [33], specifically a bi-directional flow in a constrained environment, whose scene setup is shown in Fig. 11. In this experiment, two groups of pedestrians travel down a narrow corridor $3.6\,\mathrm{m}$ wide in opposing directions. Though this presents a restricted environment, agents attempt to move in a constant direction due to which they can lookahead, and their future position can be determined with some reasonable accuracy. We compare the behavior of real pedestrians with that of simulated crowds first by examining the relationship between speed and density (commonly known as the fundamental diagram), and second, by using the metric defined in section 4.2.1. We provide comparison results using the discrete collision avoidance algorithm, RVO2 [25], [36]. Fig. 12 demonstrates how adding lookahead causes the simulated agents to exhibit a density-dependent behavior similar to that of real pedestrians. Real pedestrians demonstrate a significant sensitivity to density. Speeds decrease by a factor of 2–3 as crowd density increases for 1 person per m^2 to 3 people per m^2. However, a significant number of agents using RVO2 move with a maximum speed towards the goal as denoted by the number of sample points at $v = 1.6\mathrm{m/s}$ in Fig. 12 (Top). In Fig. 12 (Bottom), we note that agents show the same downward trend of speed with increasing density as real pedestrians. Though the observed patterns do not match human behavior exactly, the graphs demonstrate that adding lookahead

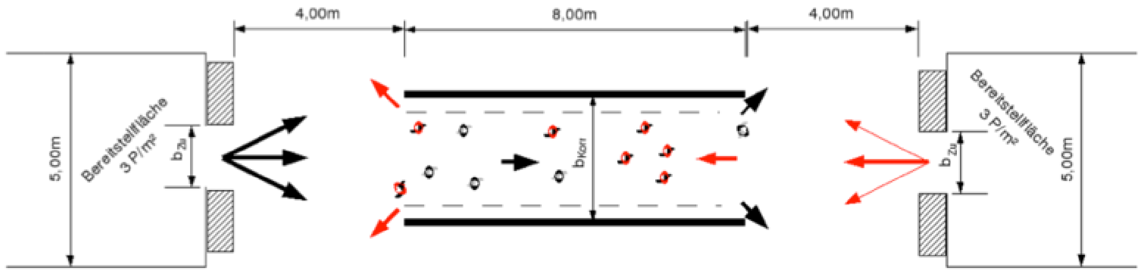
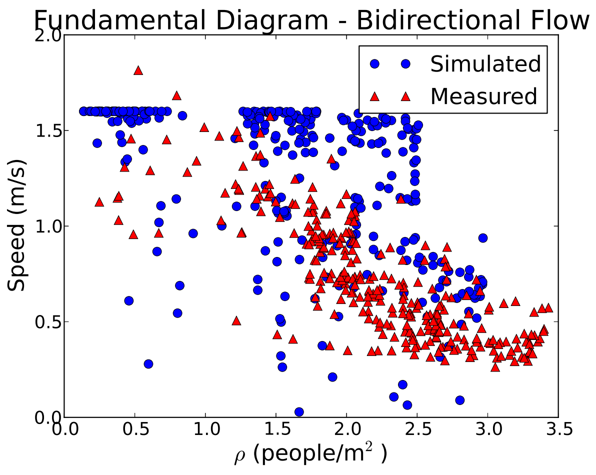


Fig. 11. Experimental setup for bi-directional crowd flow (Image courtesy [33]).



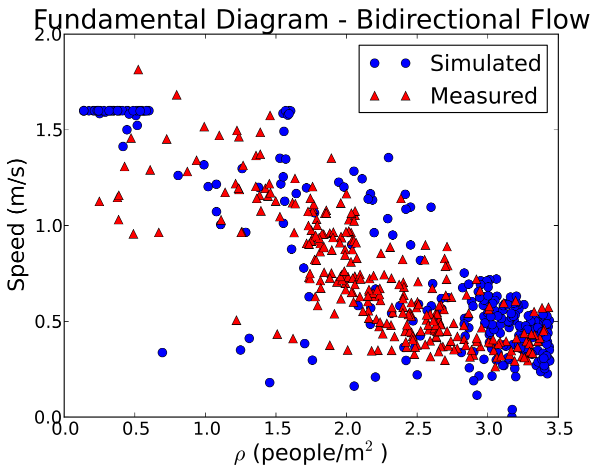


Fig. 12. Speed v.s. density plots for simulated crowds using (Top) traditional local collision avoidance, and (Bottom) long-range collision avoidance. Note how simulation with lookahead improves correspondence to speeds observed in the real-world data as compared to density, and how collision avoidance with lookahead demonstrates the same downward trend in speed with increasing density.

improves the behavior of simulated agents. Exact matching cannot be expected unless the underlying local collision avoidance algorithm models human behavior exactly. One caveat is that agents exhibit a decreased aversion to being in dense scenarios, as is indicated by significant clustering in density range of 3–3.5 people per m^2. This is partly due to lower speeds resulting in slower dissipation of congestion.

Simulated crowds in the experimental setup get to their goals in 98.4 seconds with lookahead, while taking approximately 169.88 seconds with purely local collision avoidance. For the same example, real crowds are able to navigate the same scene in 76.75 seconds. Thus, in addition to improved crowd behavior, lookahead results in a 1.73x improvement in crowd progress bringing the progress of the simulated crowd within 28.2% of observed crowds. Using a window of 10 time steps, i.e. 0.4 seconds, we computed our inconsistency metric for simulated crowds. Local collision avoidance produced metric values in the range $[0, 3.192]$ with a mean of 0.635, while crowds with lookahead resulted in values in the range $[0, 2.776]$ with a mean of 0.505. The inconsistency metric measures path smoothness as indicated by low-acceleration. By this measure, lookahead improves the mean path smoothness by 20.55%. Real-world observations in this experimental setup are made using head-trackers, which introduce a minor oscillation into agent positions. To remove this oscillation, we compute metric values for real-world data after smoothing the trajectories over a window of 2 time steps. The resulting data produces metric values in the range $[0, 2.213]$ with a mean of 1.45.

8 Conclusions and Future Work

We have presented a new, generic algorithm that can extend both existing discrete and continuum methods to provide a simple yet effective lookahead to achieve long-range collision avoidance for crowd simulations. This approach results in smoother crowd movement and exhibits an agent's tendency to avoid congestion that is often observed in real crowds. To quantify the smoothness of agent trajectories, we propose a novel metric. This metric also serves as the means to curtail the extend of lookahead in presence of chaotic crowd

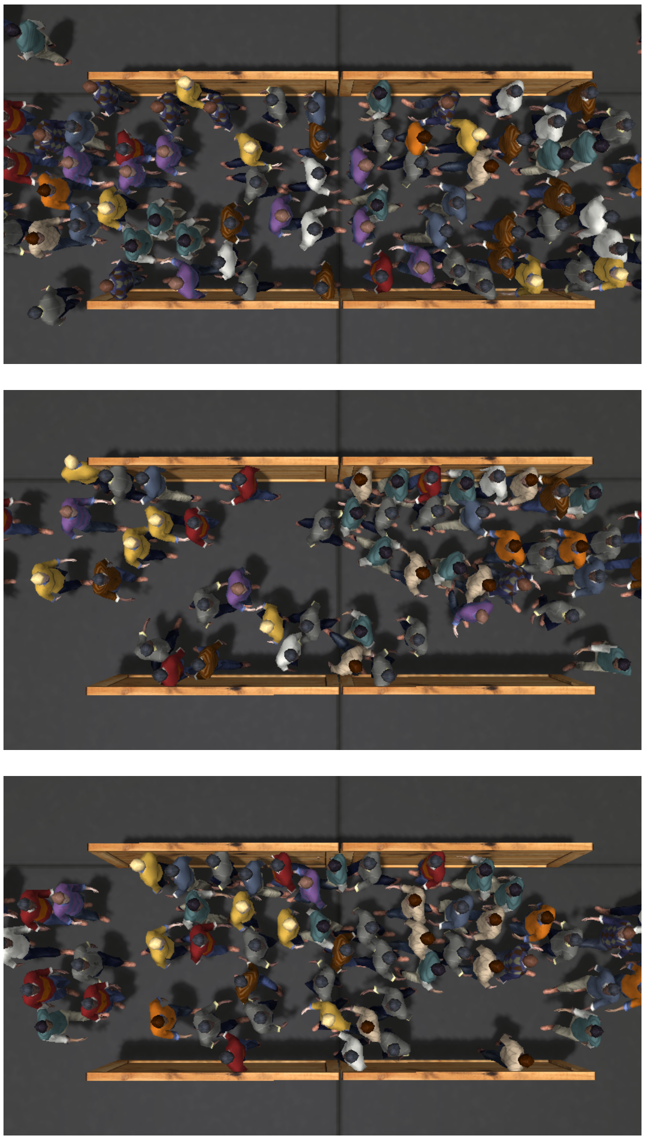


Fig. 13. Crowd motion using (Top) real-world data, (Middle) Local collision avoidance, (Bottom) Long-range collision avoidance.

behavior. In addition, we propose the use of this metric for comparing crowd simulation algorithms based on smoothness, as well as comparing simulated crowds to real-world data. We have further introduced a hybrid technique that enables the simulation system to seamlessly transition between discrete and continuum formulations by locally blending the results and by optimizing for performance and quality of resulting simulations based on the local crowd density.

Our stated goal is to improve crowd flow using long-range collision avoidance; measured as crowd progress. With the improvements proposed in this paper, simulated crowds reach their desired goals 1.5x-2.6x faster (shown in Table 1) while using speeds similar to those used by real crowds in similar scenarios (shown in Fig. 12). As shown by these measures, lookahead brings the behavior of simulated crowds closer to real humans. However, in some cases we observe outliers. After detailed analysis, we believe these cases arise due to the underlying collision avoidance model. In case of Fig. 12, solitary agents sometimes turn around to avoid collisions with oncoming groups of agents. This occurs when the space behind said agent is empty. Thus the geometrically optimal solution for collision avoidance as defined by [25] led the agent to turn around. This behavior is independent of lookahead and inherent in the underlying collision avoidance algorithm. However, lookahead allows agents behind such agents to preemptively avoid the oncoming group. Though this improves overall flow, it makes this artifact visually more prominent.

In case of Fig. 1, the larger group parts cleanly for the other. The underlying [36] algorithm optimizes with respect to hard constraints. Thus, constraints from near agents and far groups are equally important and must be respected. In this case, it leads to the clean separation of the larger group. To remedy this artifact, the formulation of constraints in [25] would need to be revisited taking into account of observed human behaviors.

Acknowledgements: This research was supported in part by ARO Contract W911NF-04-1-0088, NSF awards 0917040, 0904990, 100057 and 1117127, and Intel.

References

[1] J. Ondřej, J. Pettré, A.-H. Olivier, and S. Donikian, "A synthetic-vision based steering approach for crowd simulation," in *ACM SIGGRAPH 2010 papers*, ser. SIGGRAPH '10. New York, NY, USA: ACM, 2010, pp. 123:1–123:9.

[2] R. Narain, A. Golas, S. Curtis, and M. C. Lin, "Aggregate dynamics for dense crowd simulation," *ACM Trans. Graph.*, 2009.

[3] W. Shao and D. Terzopoulos, "Autonomous pedestrians," *Graph. Models*, vol. 69, no. 5-6, pp. 246–274, 2007.

[4] O. B. Bayazit, J.-M. Lien, and N. M. Amato, "Better group behaviors in complex environments with global roadmaps," *Proc. 8th Intl. Conf. Artificial Life*, pp. 362–370, 2002.

[5] A. Kamphuis and M. Overmars, "Finding paths for coherent groups using clearance," *Proc. of ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pp. 19–28, 2004.

[6] F. Lamarche and S. Donikian, "Crowd of virtual humans: a new approach for real-time navigation in complex and structured environments," *Computer Graphics Forum*, vol. 23, no. 3, pp. 509–518, 2004.

[7] J. Pettré, J.-P. Laumond, and D. Thalmann, "A navigation graph for real-time crowd animation on multilayered and uneven terrain," *First Intl. Workshop on Crowd Simulation*, pp. 81–90, 2005.

[8] M. Sung, M. Gleicher, and S. Chenney, "Scalable behaviors for crowd simulation," *Computer Graphics Forum*, vol. 23, no. 3 (Sept), pp. 519–528, 2004.

[9] A. Sud, R. Gayle, E. Andersen, S. Guy, M. Lin, and D. Manocha, "Real-time navigation of independent agents using adaptive roadmaps," in *Proc. ACM Symp. Virtual Reality Software and Technology*, 2007, pp. 99–106.

[10] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," *ACM SIGGRAPH*, vol. 21, pp. 25–34, 1987.

[11] D. Helbing and P. Molnár, "Social force model for pedestrian dynamics," *Physical Review E*, vol. 51, p. 4282, May 1995.

[12] C. W. Reynolds, "Steering behaviors for autonomous characters," *Game Developers Conference*, vol. 1999, 1999.

[13] L. Heigeas, A. Luciani, J. Thollot, and N. Castagné, "A physically-based particle model of emergent crowd behaviors," *Proc. Graphikon '03*, vol. 2, 2003.

[14] T. I. Lakoba, D. J. Kaup, and N. M. Finkelstein, "Modifications of the Helbing-Molnar-Farkas-Vicsek social force model for pedestrian evolution," *SIMULATION*, vol. 81, p. 339, 2005.

[15] Y. Sugiyama, A. Nakayama, and K. Hasebe, "2-dimensional optimal velocity models for granular flows," in *Pedestrian and Evacuation Dynamics*, 2001, pp. 155–160.

[16] N. Pelechano, J. M. Allbeck, and N. I. Badler, "Controlling individual agents in high-density crowd simulation," *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 99–108, 2007.

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

[18] F. Feurtey, "Simulating the collision avoidance behavior of pedestrians," Master's thesis, University of Tokyo, School of Engineering, Feb. 2000.

[19] S. Paris, J. Pettre, and S. Donikian, "Pedestrian reactive navigation for crowd simulation: a predictive approach," *Computer Graphics Forum*, vol. 26, no. 3, pp. 665–674, September 2007.

[20] A. Sud, E. Andersen, S. Curtis, M. Lin, and D. Manocha, "Real-time path planning in dynamic virtual environments using multi-agent navigation graphs," *IEEE Trans. Visualization and Computer Graphics*, vol. 14, no. 3, pp. 526–538, 2008.

[21] M. Kapadia, S. Singh, W. Hewlett, and P. Faloutsos, "Egocentric affordance fields in pedestrian steering," in *Proceedings of the 2009 symposium on Interactive 3D graphics and games*, ser. I3D '09. New York, NY, USA: ACM, 2009, pp. 215–223.

[22] J. van den Berg, M. C. Lin, and D. Manocha, "Reciprocal velocity obstacles for realtime multi-agent navigation," *Proc. IEEE Conf. Robotics and Automation*, pp. 1928–1935, 2008.

[23] J. van den Berg, S. Patil, J. Sewall, D. Manocha, and M. C. Lin, "Interactive navigation of individual agents in crowded environments," *Proc. ACM Symposium on Interactive 3D Graphics and Games*, pp. 139–147, 2008.

[24] S. Guy, J. Chhugani, C. Kim, N. Satish, M. C. Lin, D. Manocha, and P. Dubey, "Clearpath: Highly parallel collision avoidance for multi-agent simulation," *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2009.

[25] J. van den Berg, S. J. Guy, M. C. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," *Proc. Intl. Symposium on Robotics Research*, 2009.

[26] S. J. Guy, J. Chhugani, S. Curtis, P. Dubey, M. Lin, and D. Manocha, "Pledestrians: a least-effort approach to crowd simulation," in *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA '10. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2010, pp. 119–128.

[27] A. Treuille, S. Cooper, and Z. Popovic, "Continuum crowds," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 1160–1168, 2006.

[28] S. Singh, M. Kapadia, P. Faloutsos, and G. Reinman, "Steerbench: a benchmark suite for evaluating steering behaviors," *Computer Animation and Virtual Worlds*, vol. 20, no. 5-6, pp. 533–548, 2009.

[29] U. Weidmann, "Transporttechnik der fussgaenger," ETH Zürich, Tech. Rep. 90, 1993.

[30] S. J. Guy, J. van den Berg, W. Liu, R. Lau, M. C. Lin, and D. Manocha, "A statistical similarity measure for aggregate crowd dynamics," *ACM Trans. Graph.*, vol. 31, no. 6, pp. 190:1–190:11, Nov. 2012. [Online]. Available: http://doi.acm.org/10.1145/2366145.2366209

[31] A. Seyfried, B. Steffen, W. Klingsch, and M. Boltes, "The fundamental diagram of pedestrian movement revisited," *J. Stat. Mech.*, no. 10, October 2005.

[32] J. Zhang, W. Klingsch, A. Schadschneider, and A. Seyfried, "Transitions in pedestrian fundamental diagrams of straight corridors and t-junctions," *J. Stat. Mech.*, vol. 2011, no. 06, p. P06004, 2011.

[33] J. Zhang, W. Klingsch, A. Schadschneider, and A. Seyfried, "Ordering in bidirectional pedestrian flows and its influence on the fundamental diagram," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2, p. 2, Feb. 2012.

[34] A. Seyfried, O. Passon, B. Steffen, M. Boltes, T. Rupprecht, and W. Klingsch, "New insights into pedestrian flow through bottlenecks," *Transportation Science*, pp. 395–406, 2009.

[35] J. Fruin, *Pedestrian planning and design*. Metropolitan Association of Urban Designers and Environmental Planners, 1971.

[36] J. van den Berg, S. J. Guy, J. Snape, M. C. Lin, and D. Manocha. RVO2 Library: Reciprocal collision avoidance for real-time multi-agent simulation.

[37] R. L. Hughes, "The flow of human crowds," *Annual Review of Fluid Mechanics*, vol. 35, no. 1, pp. 169–182, 2003.

Abhinav Golas received a BTech & MTech in Computer Science and Engineering from the Indian Institute of Technology, Delhi in 2008. He is currently a Ph.D. candidate of computer science at the University of North Carolina at Chapel Hill. His research interests include computer graphics, physically-based modeling and simulation, and architecture-aware algorithms. His current research relates to large-scale flow simulations of fluids and crowds.

Rahul Narain received a B.Tech. in Computer Science and Engineering from the Indian Institute of Technology Delhi, and a Ph.D. in Computer Science from the University of North Carolina at Chapel Hill. He is a post-doctoral scholar at the University of California, Berkeley. His research interests include simulation of fluids, crowds, cloth, and other dynamical systems.

Sean Curtis Sean Curtis received the BA degree in German from Brigham Young University, the BS degree in Computer Science from the University of Utah and the MS in Computer Science from the University of North Carolina at Chapel Hill where he is working towards his PhD. His research interests include pedestrian modeling, multi-agent navigation, motion synthesis, crowd visualization and collision detection.

Ming Lin received her B.S., M.S., and Ph.D. degrees in Electrical Engineering and Computer Science from the University of California, Berkeley. She is current the John R. & Louise S. Parker Distinguished Professor of Computer Science at the University of North Carolina (UNC), Chapel Hill and an honorary Chair Professor (Yangtze Scholar) at Tsinghua University in China. She has received several honors and awards, including IEEE VGTC VR Technical Achievement Award 2010, and 9 best paper awards. She is a Fellow of ACM and IEEE. Her research interests include physically-based modeling, real-time interactive 3D graphics, virtual environments, geometric modeling, and GPU-Computing. She has (co-) authored more than 250 refereed publications and coedited/ authored four books. She is the Editor-in-Chief of IEEE Transactions on Visualization and Computer Graphics and a member of six editorial boards of scientific journals. She has served as a program/paper committee member for over 120 leading conferences and co-chaired over 25 international conferences and workshops.