Hybrid Long-Range Collision Avoidance for Crowd Simulation

Abhinav Golas*[1], Rahul Narain†[2], and Ming Lin‡[1]

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

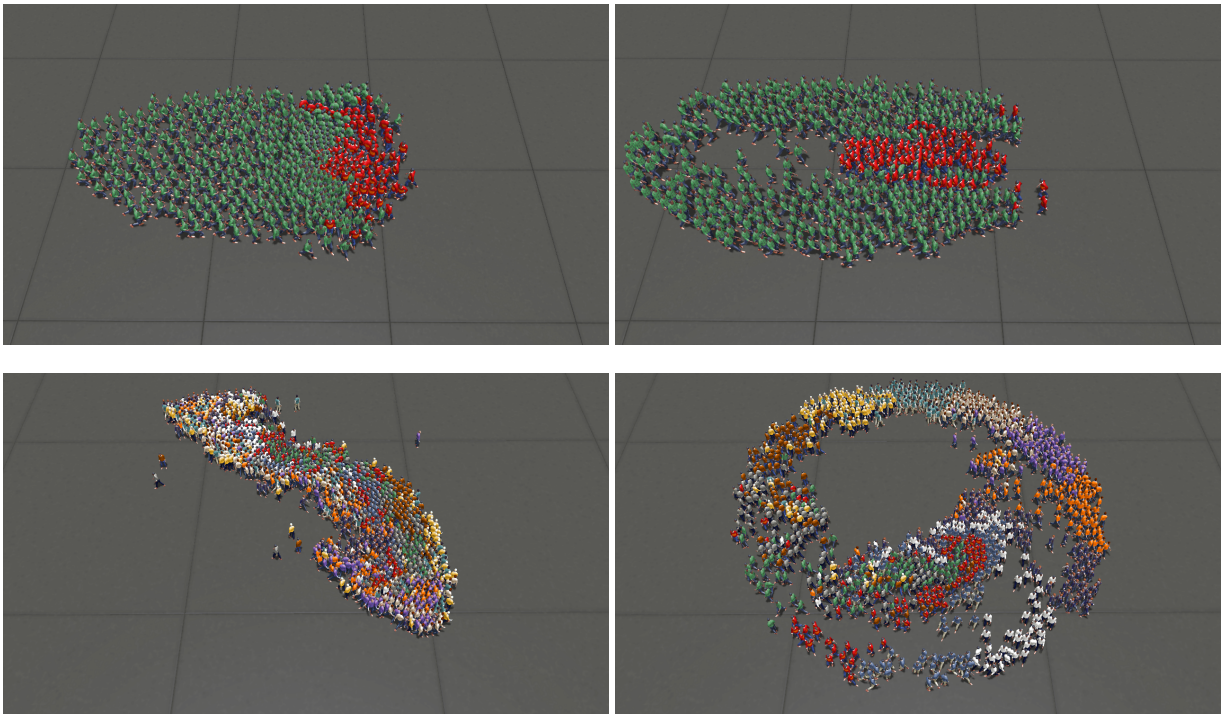


Figure 1: *Results with no lookahead (left) and lookahead (right) for 2 demo scenarios.* ***Crossing****: (Top) shows two groups of agents heading to diametrically opposite ends $t = 10s$, note how bigger group parts to allow smaller group through with lookahead.* ***Circle****: (Bottom) shows agents on the edge of a circle heading to diametrically opposite points $t = 40s$, note significantly improved progress with lookahead*

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.

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.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling

Keywords: 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 consider-

*golas@cs.unc.edu
†narain@eecs.berkeley.edu
‡lin@cs.unc.edu

ing the distribution of other pedestrians and obstacles over a large distance, people can anticipate overcrowded regions and navigate around them, thereby finding an efficient, uncongested path to their goals. 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.

The state of the art for this topic is a synthetic-vision based steering algorithm proposed by Ondřej et al. [2010]. 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 collision avoidance algorithms to provide long-range look-ahead.

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 look-ahead 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 of agent density. Continuum algorithms (e.g. [Narain et al. 2009]) are ideally suited for medium to high densities, since the continuum assumption does not hold for sparse crowds. 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. This insight suggests the need for an inexpensive hybrid scheme that blends both approaches for efficient crowd simulation over the entire spectrum of crowd densities. Such a scheme should choose the most suitable underlying algorithm for a particular simulation domain, especially for problem cases where either approach can likely cause instabilities or other computational issues 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
- A hybrid algorithm that utilizes existing continuum and discrete collision avoidance algorithms to efficiently compute smooth local collision avoidance responses

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 scene, but also achieve interactive rates on a mobile platform.

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.

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 [Funge et al. 1999; Bayazit et al. 2002; Kamphuis and Overmars 2004; Lamarche and Donikian 2004; Pettré et al. 2005; Sung et al. 2004; Sud et al. 2007]. 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 [Heigeas et al. 2003; Lakoba et al. 2005; Sugiyama et al. 2001; Sud et al. 2007], or geometrically based algorithms [Fiorini and Shiller 1998; Feurtey 2000; Sud et al. 2008]; the state of the art involves treating possible collisions as obstacles in velocity space [van den Berg et al. 2008a; van den Berg et al. 2008b; Guy et al. 2009; van den Berg et al. 2009]. 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.

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. [2006] 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. [2009] 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.

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. Further, using discrete models alone can be significantly expensive in dense crowds, while continuum models are poorly suited to representing the motion of sparse agents. In section 4, 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 5, and offer some conclusions and avenues for future work in section 6.

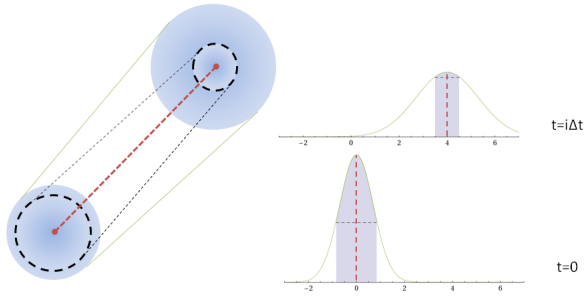


Figure 2: *Effect of extrapolation in time from $x = 0, t = 0$ to $x = 4, t = i\Delta t$. Dotted line indicates effective radius ($P \geq 0.4$) of agent for collision avoidance, while spread of gaussian curve indicates splatting area for density field creation*

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 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 travelling "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 an 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 $\Delta t_i = 2^{i-1}\Delta t_1$. 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 overriden if needed to avoid imminent collisions with nearby agents, which are considered later in the process.

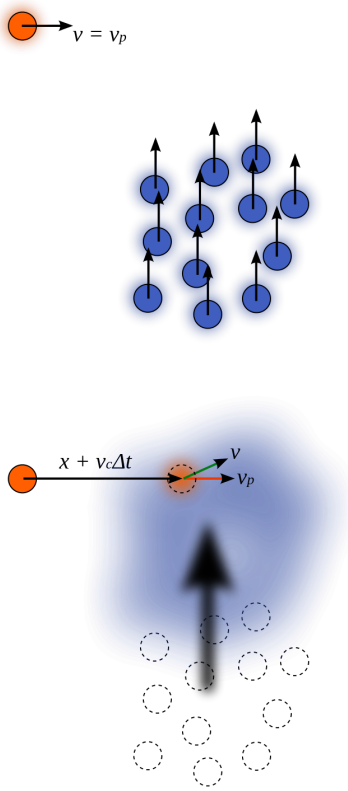


Figure 3: *Effect of lookahead. Note how lookahead allows the orange agent to see the approaching crowd and adjust its velocity from preferred velocity v_p to v by incorporating information from the future crowd state at time $t + \Delta t$*

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, producing a collision-free velocity v. In the following

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

Figure 4: *Lookahead Algorithm using LCA algorithm A*

subsections, we apply 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 [Fruin 1971], a fact that can be formalized into a graph known as the *fundamental diagram* relating density, ρ, to maximum 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 farther 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

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

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,

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

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 [Narain et al. 2009], 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 Figure 4 can be directly applied. For each cell at the bottom of the pyramid, we solve the LCA problem at multiple levels of the pyramid, starting from the top and cascading the solution at level i as the preferred velocity at level $i - 1$. We prefer this approach as opposed to doing complete solves for all locations at level i and interpolating changes down. This allows a completely parallel computation, as well as reduction in interpolation and smoothing artifacts.

3.2 Discrete Lookahead

We now extend the algorithm to discrete collision avoidance. Here, we use the reciprocal velocity obstacle (RVO) algorithm [van den Berg et al. 2009], which is implemented in the RVO2 library [van den Berg et al.]. 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 traditional RVO algorithm requires the choice of a neighborhood radius R and time horizon τ, and only considers collisions 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 and stability 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 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 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 farther in time, their effective radius reduces further, and thus have a decreased effect on agent velocity.

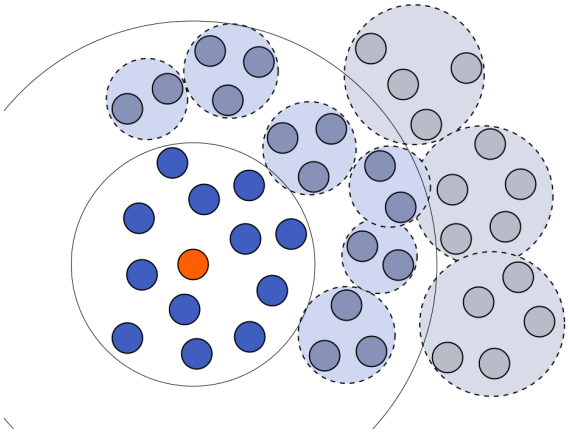


Figure 5: *Distant agents can be clustered for collision avoidance, cluster size being proportional to distance. Since possible collisions with distant agents lie in future timesteps, extrapolated future agent states have high uncertainty, and hence small effective radii, making individual avoidance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

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 kD tree, over the agent positions to choose the clusters. Typically, 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 figure 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 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

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

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

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

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.

Figure 6: *Lookahead Algorithm using RVO*

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

4 Hybrid Crowd Simulation

Collision avoidance guarantees, provided by algorithms discussed this far, are conditional on certain assumptions. In situations where these assumptions are violated, collision avoidance guarantees do not hold, and the resulting artifacts can produce incorrect or at least visual 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, this is reflected in the shrinking of the solution space, meaning that the algorithm needs to spend more computational time to converge to this region, 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:

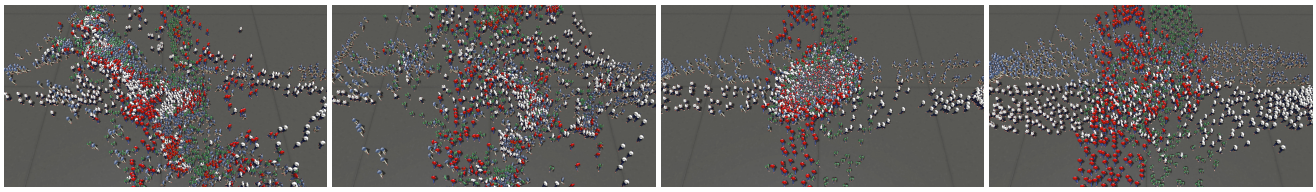
- $[0, \rho_{cmin}]$: Discrete collision avoidance
- $[\rho_{cmin}, \rho_{dmax}]$: Blend Discrete and Continuum collision avoidance
- $[\rho_{dmax}, \rho_{max}]$: Continuum collision avoidance

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:

$$v = v_{disc}(1 - w) + v_{cont}w \tag{6}$$

$$w = w_\rho = clamp\left(\frac{\rho - \rho_{cmin}}{\rho_{dmax} - \rho_{cmin}}, 0, 1\right) \tag{7}$$

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.



(a) (b) (c) (d)

Figure 7: ***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	Method	Time per Step (ms)
Crossing	500	Disc	0.98
		Disc LA	3.68
Circle	1000	Disc	2.6
		Hyb LA	5.2
4 way	2000	Disc	5
		Disc LA	47.3
		Cont LA	6.8
		Hyb LA	16.29
4 groups	2000	Cont	6.6
		Cont LA	6.89

Table 1: *Single thread performance for our examples ($dt = 0.01s$).* ***Legend****: **LA** - with lookahead, **Disc** - Discrete, **Cont** - Continuum, **Hyb** - Hybrid*

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 (7), we can define blending weights for this case as well:

$$w_{\sigma_v} = clamp\left(\frac{c_2 v - \sigma_v}{(c_2 - c_1)\, v}, 0, 1\right) \tag{8}$$

Note that this weight has a slightly different form to maintain the convention in (6).

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 [Narain et al. 2009] and others, velocity variance decreases at high densities. Thus adding a weight for velocity variance does not affect high density behavior significantly.

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

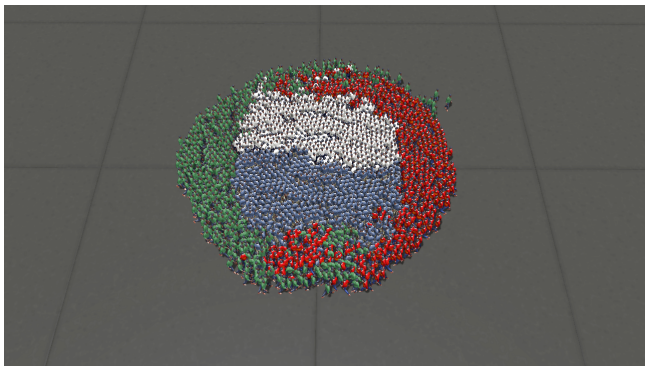
Our lookahead and hybrid algorithms were tested on a number of cases. The first example scenes demonstrate the lookahead algorithm for the continuum and discrete cases. In the discrete case, two groups of agents - one bigger than the other - head towards each other on a collision course. Using our lookahead algorithm, the bigger 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 the double, thus over the time of the simulation, the overall cost is less than 2x In the continuum case 8, where 4 groups of agents attempt to reach diametrically opposite regions, agents avoid the high density region in the center. As compared 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 figure 7 shows the difference in the 4 kinds of simulation.

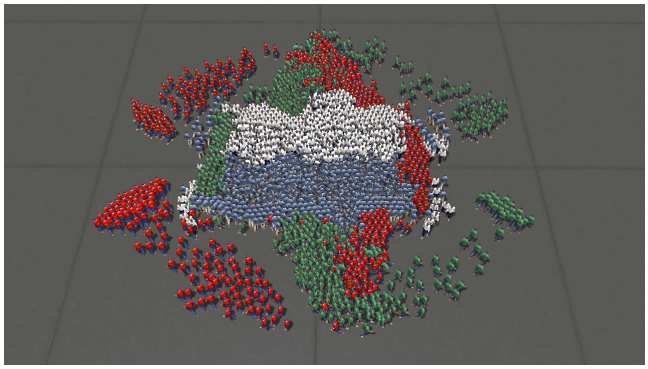
Lastly, we replicated the well known circle demo with 2000 agents. In this scenario, agents are seeded on a circle and attempt to reach the diamterically opposite point. At a 2X extra computational cost, we observe significant improvement in behavior and progress. Agents are able to reach their goal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

6 Conclusion

We have present 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. We have further introduced a hybrid technique that enables the simulation system to seamlessly transition between discrete and continuum formulations by blending the results between regions and by optimizing for performance and quality of resulting simulations based on the local crowd density.



(a)



(b)

Figure 8: ***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)*

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

References

BAYAZIT, O. B., LIEN, J.-M., AND AMATO, N. M. 2002. Better group behaviors in complex environments with global roadmaps. *Proc. 8th Intl. Conf. Artificial Life*, 362–370.

FEURTEY, F. 2000. *Simulating the Collision Avoidance Behavior of Pedestrians*. Master's thesis, University of Tokyo, School of Engineering.

FIORINI, P., AND SHILLER, Z. 1998. Motion planning in dynamic environments using velocity obstacles. *Intl. J. on Robotics Research 17*, 7, 760–772.

FRUIN, J. 1971. *Pedestrian planning and design*. Metropolitan Association of Urban Designers and Environmental Planners.

FUNGE, J., TU, X., AND TERZOPOULOS, D. 1999. Cognitive modeling: Knowledge, reasoning and planning for intelligent characters. *Proc. of ACM SIGGRAPH*, 29–38.

GUY, S., CHHUGANI, J., KIM, C., SATISH, N., LIN, M. C., MANOCHA, D., AND DUBEY, P. 2009. Clearpath: Highly parallel collision avoidance for multi-agent simulation. *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*.

HEIGEAS, L., LUCIANI, A., THOLLOT, J., AND CASTAGNÉ, N. 2003. A physically-based particle model of emergent crowd behaviors. *Proc. Graphikon '03 2*.

KAMPHUIS, A., AND OVERMARS, M. 2004. Finding paths for coherent groups using clearance. *Proc. of ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 19–28.

LAKOBA, T. I., KAUP, D. J., AND FINKELSTEIN, N. M. 2005. Modifications of the Helbing-Molnar-Farkas-Vicsek social force model for pedestrian evolution. *SIMULATION 81*, 339.

LAMARCHE, F., AND DONIKIAN, S. 2004. Crowd of virtual humans: a new approach for real-time navigation in complex and structured environments. *Computer Graphics Forum 23*, 3, 509–518.

NARAIN, R., GOLAS, A., CURTIS, S., AND LIN, M. C. 2009. Aggregate dynamics for dense crowd simulation. *ACM Trans. Graph.*.

ONDŘEJ, J., PETTRÉ, J., OLIVIER, A.-H., AND DONIKIAN, S. 2010. A synthetic-vision based steering approach for crowd simulation. In *ACM SIGGRAPH 2010 papers*, ACM, New York, NY, USA, SIGGRAPH '10, 123:1–123:9.

PETTRÉ, J., LAUMOND, J.-P., AND THALMANN, D. 2005. A navigation graph for real-time crowd animation on multilayered and uneven terrain. *First Intl. Workshop on Crowd Simulation*, 81–90.

SUD, A., GAYLE, R., ANDERSEN, E., GUY, S., LIN, M., AND MANOCHA, D. 2007. Real-time navigation of independent agents using adaptive roadmaps. In *Proc. ACM Symp. Virtual Reality Software and Technology*, 99–106.

SUD, A., ANDERSEN, E., CURTIS, S., LIN, M., AND MANOCHA, D. 2008. Real-time path planning in dynamic virtual environments using multi-agent navigation graphs. *IEEE Trans. Visualization and Computer Graphics 14*, 3, 526–538.

SUGIYAMA, Y., NAKAYAMA, A., AND HASEBE, K. 2001. 2-dimensional optimal velocity models for granular flows. In *Pedestrian and Evacuation Dynamics*, 155–160.

SUNG, M., GLEICHER, M., AND CHENNEY, S. 2004. Scalable behaviors for crowd simulation. *Computer Graphics Forum 23*, 3 (Sept), 519–528.

TREUILLE, A., COOPER, S., AND POPOVIC, Z. 2006. Continuum crowds. *ACM Trans. Graph. 25*, 3, 1160–1168.

VAN DEN BERG, J., GUY, S. J., SNAPE, J., LIN, M. C., AND MANOCHA, D. RVO2 Library: Reciprocal collision avoidance for real-time multi-agent simulation.

VAN DEN BERG, J., LIN, M. C., AND MANOCHA, D. 2008. Reciprocal velocity obstacles for realtime multi-agent navigation. *Proc. IEEE Conf. Robotics and Automation*, 1928–1935.

VAN DEN BERG, J., PATIL, S., SEAWALL, J., MANOCHA, D., AND LIN, M. C. 2008. Interactive navigation of individual agents in crowded environments. *Proc. ACM Symposium on Interactive 3D Graphics and Games*, 139–147.

VAN DEN BERG, J., GUY, S. J., LIN, M. C., AND MANOCHA, D. 2009. Reciprocal n-body collision avoidance. *Proc. Intl. Symposium on Robotics Research*.