17

Advanced Techniques for Robust, Efficient Crowds

Graham Pentheny

17.1 Introduction

To date, crowds in games are usually driven by static pathfinding combined with localized steering and collision avoidance. This technique is well understood and works well for small or moderate crowd sizes that are sparsely distributed in their environment. As crowd density increases and the agents' goals converge, this approach falls apart. In situations like this, agents fight the rest of the crowd to follow their ideal path as closely as possible, seemingly ignoring all other possible routes.

Congestion maps provide a simple means of modeling aggregate behavior in large crowds of tens or hundreds of thousands of agents. Combined with *vector flow fields*, congestion maps can be used to elegantly handle large variations in path travel times due to congestion and crowding from other agents in the scene. Agents controlled by this technique appear to be more aware of their surroundings, reconsidering their current path choice if a less-congested alternative exists.

17.2 Pathfinding's Utopian Worldview

Current solutions for pathfinding compute paths in an idealized environment. They compute the shortest distance path from the current agent position to its goal through a given environment as if no other agents existed. Some variations of common pathfinding approaches such as Partial Refinement A* [Sturtevant 05] repeatedly perform partial path calculations to account for changes in the environment. This is ideal for rapidly changing environments as much of a full path calculation is likely to never be used before it is necessary to repath. Despite adeptly handling changes to the open areas in the environment, Partial Refinement variants of A* operate on the principle that the only obstacles in the environment are static and binary. Either the agent can move through a given location or it can't. These approaches cannot handle situations where certain path directions are obstructed by large groups of other agents, because they do not consider other agents in their calculations. This can be thought of as optimizing for total path length, rather than for the actual travel time. In an empty environment, the travel time is simply the distance inversely scaled by the agent's velocity. However, in a crowded world, other agents increase the travel time through certain points in the environment.

Collision avoidance algorithms attempt to avoid other agents in the local vicinity by either restricting or augmenting the agent's velocity. These techniques are well suited for avoiding localized collisions and are often used to complement idealized pathfinding. Collision avoidance algorithms, however, fail to account for situations where movement is significantly impacted by other agents. Collision avoidance helps avoid other agents that may obstruct travel along the ideal, shortest path, but does not deviate the path direction. In games using this dual-layered pathing and collision avoidance approach, repathing occurs only when the collision avoidance solver pushes the agent far enough away from their current, ideal path so that it no longer remains the ideal movement direction. The implications of this are most easily seen in the following example, illustrated in Figure 17.1.

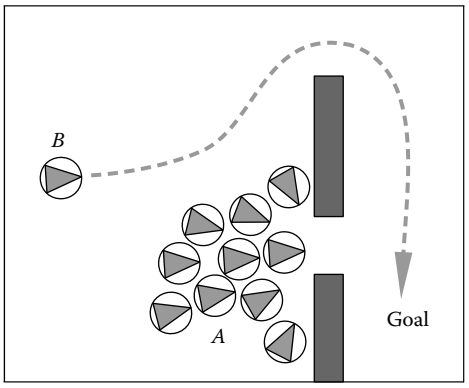

Figure 17.1

Two groups of agents attempting to reach a shared goal. The dashed line indicates the time-optimal path for agent *B*.

In this scenario, the agents are attempting to reach a common goal. There are a large number of agents in group *A*, and therefore they will take a considerable amount of time to move through the lower opening. Because all agents follow their ideal shortest path, both groups attempt to move through the lower opening. The agent marked as "*B*" in this example could path through the upper unobstructed opening, resulting in a shorter travel time to its goal. This alternative path is indicated as a gray dashed line in Figure 17.1. With just path planning and collision avoidance, agent *B* will consider the alternative path only when its collision avoidance moves the agent close to the upper opening. Only then, when the agent's shortest path is through the upper opening, will it consider this alternative route.

Both approaches that compute individual paths, and those like flow fields that aggregate pathfinding information across a group, suffer from this lack of contextual awareness. These algorithms are designed to minimize the total path distance, as this is a measurable and easily optimized metric. Alternatively, full movement-planning approaches involve predicting and planning movement for all agents through the scene and solving for collisions and congestion over their entire respective paths. These approaches can be too resource intensive to compute for large numbers of agents in real time. Additionally, motion planning does not account for future environmental or goal changes that may affect the computed movement information. As such, motion planning is only seen in situations requiring high-quality movement for small numbers of agents.

17.3 Congestion Map Approach

The solution we propose is a hybrid of static, idealized pathfinding, with a simplification of motion planning. The approach computes aggregate crowd density and movement across the environment and then compares that to the idealized path direction and movement. This information, called a *congestion map*, is then used to augment pathfinding computations by discouraging movement through crowded areas. Congestion maps offer significant advantages over *direction maps* (DMs) [Jansen 08], an approach we will compare later in this chapter.

This approach is based on the observation that moving against or across a crowd is much more difficult than movement aligned with the flow of the crowd. Additionally, the more densely crowded an area is, the more time it will cost to move through.

Because information on crowd dynamics is computed for groups of agents as a whole, the computational complexity is far less than that of full motion planning. Combined with aggregate pathfinding techniques such as flow fields, this approach scales well to large numbers of agents, making it ideal for large crowd simulations.

17.4 Augmenting Path Planning with Congestion Maps

The first step in the process is to compute the aggregate crowd density across the environment. For each agent in the scene, we record its current velocity and position. We then use the agent positions to construct an agent influence map [Champandard 11] across the environment space. This is used as an estimation of agent density, where larger influence values correspond to a denser crowd at that position. Conversely, lower values indicate more sparsely distributed agents.

Velocity information is plotted and distributed over the environmental space in the same manner. In addition, we compute a rolling average of the agent velocity vectors, providing a map of the average velocity of the crowd at each point. For clarity, both the average velocity and the crowd density information are referred to collectively as a "congestion map." The congestion map indicates areas that are likely to experience increased travel costs due to crowd congestion.

The congestion map information is then used as a means of computing path traversal cost in a heuristic pathfinding algorithm, such as A*. Crowd density alone could be interpreted as a traversal cost; however, this would cause agents moving together at a uniform velocity to unnecessarily avoid each other. Instead, we use the aggregate crowd velocity information to ensure that we only add traversal costs when necessary. Using an unmodified pathfinding algorithm, we begin to compute the ideal path from the agent to the goal. In the pathfinding process, we use the congestion map to augment the computed traversal cost from one step in the path to the next. This is done by computing the difference in the aggregate crowd velocity and the agent's ideal path velocity. The scalar magnitude of this vector difference is then scaled by the crowd density value, resulting in the "congestion penalty" for that movement. Pseudocode for computing the traversal cost is shown in Listing 17.1. The congestion penalty can then be easily integrated into existing calculations as an additional path traversal cost. Furthermore, because the crowd congestion penalty is never negative, it maintains heuristic admissibility in the path planner.

As agents move through the scene, they use the congestion map to augment their pathfinding queries. This ensures that they will favor paths that offer the shortest travel time, even if they are not the shortest in distance. Each agent performs a traditional pathfinding query, or an aggregate pathfinding pass is performed in the case of using flow fields. The normal behavior of the heuristic path planner uses the "congestion map" information to choose paths with lower costs and thus minimal travel times. Alternatively, if a congested path still remains better than any alternative, the path planner will correctly choose the congested path.

Listing 17.1. Traversal cost computation using the congestion map information.

```
float congestionPenalty(Vec2 ideal,
                        Vec2 aggregate,
                        float density)
{
    //Projection of aggregate onto ideal, represented
    //as a scalar multiple of ideal.
    float cost = Vec2.dot(ideal, aggregate);
    cost /= ideal.mag() * ideal.mag();

    //If cost is > 1, the crowd is moving faster along the
    //ideal direction than the agent's ideal velocity.
    if (cost >= 1) return 0.0f;

    //Cost is transformed to be positive,
    //and scaled by crowd density
    return (1 - cost) * density;
}
```

The final step in the process involves adding in local collision avoidance. While the congestion map will help deal with macrolevel collision avoidance, we still rely, albeit far less, on collision avoidance algorithms to resolve local agent collisions.

17.5 Path Smoothing

Because the congestion coefficients are used by the pathfinder as traversal cost values, unmodified path-smoothing algorithms will not maintain this information. Path smoothing relies on line-of-sight collision checks to determine whether a waypoint in a path is considered redundant and can be removed. This process creates a final, smoothed path containing the minimal number of waypoints necessary to accurately guide the agent through the static obstacles in the environment to its goal. Because the heuristic for skipping path nodes only considers line of sight, it will not produce smoothed paths that respect congestion map information.

Smoothing of paths computed with congestion map information involves comparing the movement cost of smoothed routes. Classic path-smoothing approaches assume the movement cost in the world is invariant and thus optimize for the shortest total path distance. To incorporate the congestion map information, the path-smoothing algorithm must compare the ultimate time cost of moving along both paths. To accurately compare two potential smoothed versions of a path, the smoothing algorithm must produce a heuristic that accounts for both the traversal cost and total distance of each. The heuristic estimates the total travel cost for a path by computing the line integral along path direction over the congestion penalty function. This can be easily computed over a discretized world (such as a grid) by computing the sum of each step's movement distance scaled by its corresponding traversal cost. The result of this summation (more generally of the line integral) constitutes the overall traversal cost for a path. The path-smoothing algorithm can use this value as a heuristic, allowing it to accurately compare two potential smoothed paths.

17.6 Flow Fields with Congestion Maps and Theta

In dense crowd simulations, many agents will be considering movement in a shared space. Additionally, many agents may share a set of goal destinations. As such, pathfinding approaches that exploit this uniformity across the agent population are ideal. Flow fields provide these benefits, as they unify pathfinding information for all agents with a shared goal. Flow fields compute ideal path directions for every discretized point in a given world. This provides constant computation and look-up cost for paths for any number of agents with a shared set of goals. The increased pathfinding complexity of congestion map aware algorithms amplifies the benefits of flow fields for large crowds. When using flow fields with congestion maps, the special path-smoothing considerations can be combined into the flow vector calculation process.

Flow fields are generated by back-propagating ideal path directions from the goal position using an unbounded Dijkstra's algorithm. While this is efficient and easily implemented for simple applications, it does not offer smoothed paths. Additionally, adding smoothing as a postprocess step (as in single-source pathfinding) does not scale well to large crowds due to the number of line-of-sight calculations required. These restrictions make the Theta* [Nash 07, Nash 15] algorithm ideal for generating flow fields.

Theta* operates identically to Dijkstra's algorithm when generating flow fields; however, it performs path-smoothing calculations as paths are being constructed. As Theta* creates a path link between two nodes *A* and *B*, it performs a line-of-sight check between the new node *A* and the parent of the previously expanded node *B*. This line-of-sight check then exists through the remainder of all path calculations and can be reused in subsequent path-smoothing calculations. The line-of-sight check can also incorporate congestion map information by computing and memoizing the path traversal cost via the process defined in the previous section. Theta* combined with these path cost calculations allows it to efficiently generate congestion map aware flow fields. Please see the chapter on Theta* in this book for more details on Theta* [Nash 15].

17.7 Current Alternatives

Current crowd dynamics solutions generally involve two layers: pathfinding and local collision avoidance. These approaches offer a few noteworthy benefits. They produce high-quality movement and avoidance on small scales and are well understood and researched by the community. There are many open-source and off-the-shelf implementations of these techniques, and they integrate well into existing technology. A popular choice for many games is the combination of A* with velocity obstacle [van den Berg 08] approaches such as ORCA [van den Berg 09] or ClearPath [Guy 09]. These offer an enticing combination of fast, inexpensive pathfinding with robust, high-quality collision avoidance.

In high-density crowd situations, solely relying on local collision avoidance and idealized pathfinding will cause agents to pile up at popular, shared path waypoints. Collision avoidance algorithms only help avoid local collisions in the pursuit of following the ideal path. Often games rely on these algorithms to divert agents to less-congested, less-direct routes in high-density situations. In certain situations, collision avoidance can lead to this desired behavior, though it is always a side effect of the system and not a deliberate consideration.

Work has been done in incorporating aggregate crowd movement and crowd density into pathfinding computations [van Toll 12, Karamouzas 09, Jansen 08]. Approaches that augment pathing via crowd density [van Toll 12, Karamouzas 09] do not take into account the aggregate movement or direction of movement of the crowd. This leads to overcorrection of the phenomenon illustrated in Figure 17.1.

Congestion maps are similar in many ways to existing cooperative pathfinding algorithms, such as "DMs" [Jansen 08], but differ in a few key respects. DMs use average crowd motion over time to encourage agents to move with the flow of the crowd. Because of this, many of the oscillations present in the congestion map approach are smoothly resolved. Conversely, this temporal smoothing prevents DMs from quickly and accurately reacting to changes in the environment and crowd behavior. Both congestion maps and DMs apply the aggregate crowd movement information to the path planning process in much the same way; however, congestion maps handle agents of varying size and shape, while DMs traditionally assume homogeneity. The final major difference between DMs and congestion maps is that congestion maps weight movement penalties proportional to the density of the crowd. Without taking density into account, DMs display overly pessimistic pathing behavior, where agents are encouraged to path around sparse groups of agents blocking the ideal path.

17.8 Benefits

Congestion maps offer an effective way of enhancing crowd behavior at scale. Compared to motion planning approaches that predict movement and interactions of all agents in a given time interval, congestion maps are an inexpensive addition to established character movement systems. Additionally, the simplicity of congestion maps makes them easy to implement and optimize.

Congestion maps augment agent pathfinding to work as it should. Instead of optimizing for minimal path distance, path planners using congestion maps correctly optimize for path travel time. This ensures agents will consider less-crowded alternative routes that may be slightly longer but ultimately faster than the ideal path.

Though congestion maps can be added to any existing path planning system, flow fields are ideal for exploiting the benefits of this approach. Using Theta* to generate flow fields results in drastically fewer line-of-sight checks, as their results can be shared across path calculations. Theta* minimizes the impact of the increase in path-smoothing computations with congestion maps, without reducing the technique's effectiveness.

17.9 Drawbacks

Despite the many benefits congestion maps offer, they are not a replacement for full motion planning. Congestion maps are a reactive, macrolevel collision avoidance technique. Changes to crowd density over time are not taken into account when augmenting unit paths. As such, an agent may avoid a congested area along its ideal path that, by the time the agent would reach that area, would no longer be congested. This can lead to agents appearing to "change their mind" as congestion eases in specific locations. Conversely, an agent can begin moving toward a location that is not currently congested, but that will become so once the agent reaches the area. This will cause the agent to change directions toward a longer, less-congested path. Depending on the application of the congestion map approach, these behavioral flaws may be acceptable, as they mimic the fallibility of human path planning. In other applications, their impact may be negligible.

Due to the dynamic nature of crowd density, congestion maps are best suited for highly dynamic environments and techniques. As crowd density changes, existing paths become less ideal in both distance and traversal time. This necessitates replanning existing paths to account for changes in the environment. Hierarchical discretization helps alleviate some of the costs of consistent repathing by shrinking the search space, speeding up individual pathfinding computations.

Finally, by their nature, congestion maps weaken the heuristic used for search, increasing the cost of path planning. Again, hierarchical methods or weighted A* can be used to reduce this overhead [Jansen 08].

17.10 Performance Considerations

Congestion maps compute crowd density and aggregate information across the entire environment. This requires discretizing the continuous space at some granularity. As the resolution of the congestion map data increases, the memory required to store the congestion

data also increases. Additionally, the cost of computing blended moving averages of aggregate crowd movement vectors increases with the resolution of the congestion map.

Despite holding information for every position in the environment, the congestion map doesn't need to correspond directly to the existing world discretization. In fact, the congestion map resolution can be much smaller than the world discretization resolution and still maintain much of its effectiveness. However, the coarser the congestion map resolution, the more likely agents will exhibit strange behavior, such as avoiding areas that don't need to be avoided. The overall macrolevel behavior will be correct and consistent; however, individuals may show odd patterns of behavior.

17.11 Future Work

Hysteresis can be added to the system to prevent agents from oscillating between potential paths quickly due to rapid changes in congestion information. With hysteresis, an agent will remain on its current path until the congestion values have surpassed a certain value for a certain amount of time. Likewise, the agent will not consider a shorter path until that path has been uncongested for a certain amount of time. These time intervals and congestion thresholds are user defined, offering high-level control over the behavior of the agents in the scene. Additional realism is obtained by authoring congestion coefficient levels and time delays as random distributions over specific value ranges.

Because the congestion map only offers a snapshot of the current crowd density and aggregate velocity, it is not perfectly accurate to the realities of the agents' theoretically ideal behavior. This inaccuracy is introduced as a means of improving runtime performance and simplifying implementation details. Computing the full crowd density over time would allow the path planner to more accurately compute traversal cost. With this method, the path planner can base the traversal cost on the crowd state at the time when the agent would be at the considered position in its path. This is similar to motion planning approaches, in that each agent must know the expected behavior of the other agents in the scene to compute an ideal path. Because they only require computing aggregate agent behavior, congestion maps evaluated over time intervals may also prove to be less computationally expensive than full motion planning.

17.12 Conclusion

A combination of static, idealized pathfinding and localized collision avoidance algorithms are often used to simulate crowds in games. While effective for small numbers of sparse agents, these approaches lack consideration of the effects of crowd dynamics on agents' path planning calculations.

Congestion maps introduce context awareness to the path planning system and allow individual agents to react to the agents around them on a large scale. Together with Theta*, congestion maps can generate ideal pathing information for an entire environment in the form of a vector flow field. By maximally reusing shared path computations, flow fields help reduce the cost of smoothing individually computed paths.

Adding congestion maps to a path planning system allows agents, in situations of high crowd density, to find alternative, longer paths that will ultimately take less time to follow.

This is a behavior not previously possible without expensive motion planning approaches, which provides opportunities for games to create more compelling, realistic, and interesting crowds.

References

[Champandard 11] Champandard, A. 2011. The mechanics of influence mapping: Representation, algorithm and parameters. http://aigamedev.com/open/tutorial/influence-map-mechanics/ (accessed June 1, 2014).

[Guy 09] Guy, S., Chhugani, J., Kim, C., Satish, N., Lin, M., Manocha, D., Dubey, P. 2009. ClearPath: Highly parallel collision avoidance for multi-agent simulation. *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Computer Animation* (*2009*), pp. 177–187.

[Jansen 08] Jansen, M. and Sturtevant, N. 2008. Direction maps for cooperative pathfinding. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*.

[Karamouzas 09] Karamouzas, I., Bakker, J., and Overmars, M. 2009. Density constraints for crowd simulation. *Proceedings of the ICE Games Innovations Conference*, pp. 160–168.

[Nash 07] Nash, A., Daniel, K., Koenig, S., and Felner, A. 2007. Theta*: Any-angle path planning on grids. *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 1177–1183.

[Nash 15] Nash, A. and Koenig, S. 2015. Theta* for Any-Angle Pathfinding. In *Game AI Pro²: Collected Wisdom of Game AI Professionals*, ed. S. Rabin. A K Peters/CRC Press, Boca Raton, FL.

[Sturtevant 05] Sturtevant, N. and Buro, M. 2005. Partial pathfinding using map abstraction and refinement. *Proceedings of the National Conference on Artificial Intelligence*, July 2005, Vol. 5, pp. 1392–1397.

[van den Berg 08] van den Berg, J., Lin, M., and Manocha, D. 2008. Reciprocal velocity obstacles for real-time multi-agent navigation. *IEEE International Conference on Robotics and Automation*, 1928–1935.

[van den Berg 09] van den Berg, J., Guy, S., Lin, M., and Manocha, D. 2009. Reciprocal n-body collision avoidance. *Proceedings of the International Symposium on Robotics Research*.

[van Toll 12] van Toll, W., Cook IV, A., and Geraerts, R. 2012. Real-time density-based crowd simulation. *Computer Animation and Virtual Worlds*, 23, 59–69.