Crowd Simulation

Harry Gifford[*]

Abstract

I describe an algorithm for simulating crowds in a variety of situations. I formalize the problem as a Markov Decision Process and then apply randomization techniques, along with physically and biologically based methods in order to feasibly solve this problem without discretization. I then evaluate my model with that of simple Boids flocking behavior in order to determine how successful the model is at simulating human behavior in crowds.

Keywords: crowd simulation, artificial intelligence, search, animation, many-agent simulation

1 Introduction

Given a set of agents, goals and a euclidean map how can we best simulate the movement of these agents to reach their goals?

The inputs will be a set of agents, and goal locations assigned to each agent. The output will be positions of each agent for each time period in the simulation.

As a baseline, I will use a simple Boids algorithm for simulating crowds and compare it to my algorithm. The Boids algorithm is a simplistic model for how large groups of individuals behave. The boids algorithm is a local method for determining the position and velocity for an agent based on the positions and directions of its immediate neighbors at each time step. Developed in the late 1980's, Boids simulation has been applied and extended in numerous research areas to simulate flocking behavior [Reynolds 1987] If we add a goal direction to this model it works as a reasonable algorithm for crowd simulation.

1.1 Justification

Crowd simulation is an interesting and useful subject. There are numerous uses of crowd simulation from movies to city planning. Often movies wish to animate large groups of people in ways that are either prohibitively expensive, or are physically impossible.

Architects often wish to simulate crowds in order to most efficiently design a space in a building. It is important that public spaces in buildings are well designed, so that in panic situations crowds do not add to the danger. Similarly, aircraft manufacturers must meet strict safety targets on how quickly an aircraft can be evacuated in an emergency. Therefore aircraft manufacturers would like to be able to simulate crowds to effectively guide the design of the interior of aircraft to minimize escape time [Engber 2006].

[*]e-mail:gif@stanford.edu

Public planners would like to use crowd simulation to guide the design of public transport systems, such as subways, sidewalks and highways to minimize the area and cost that these systems take up, while maximizing traffic throughput[Jund et al. 2012].

Another important use of crowd simulation is to model how psychological factors affect large groups of people, especially in panic situations [Helbing et al. 2000]. This can be useful in determining good strategies for tackling with panic situations.

1.2 Hypothesis

We will assume that agents (humans in the crowd) will attempt to optimize their time to get the the goal.

We require that our algorithm will lead to a solution, where all agents have successfully reached their goals with no collisions, if one exists.

I hypothesize that my simulation will better approximate human behavior in crowds than other methods, such as Boids simulation. In order to verify this, I will use a variety of metrics. The first, simplest evaluation will be how the average walking speed of agents through a crowd of different densities compares with speeds measured in the real world.

The second metric I will use will be the average expended energy of an agent to reach its goal. We wish the amount of energy for an agent to expend on reaching its goal to be as minimal as possible. As there is a reasonable lack of precise data on crowds (under the time constraints) I will assume that human agents in general will attempt to take paths through an arena that minimize their energy loss. This would agree with evolutionary ideas, since it is beneficial to a species to expend minimal energy when traveling. Therefore, I believe this metric is valid in evaluating our model.

The final comparison I will make is to human behavior when walking at certain speeds. This is a common test used in simulating panic situations and it is a good way to test my algorithm on crowd data that is readily available.

The benchmark I will compare my algorithm against is a simple Boids simulation, with a shortest path algorithm to guide the agents to their goals.

2 Methods

I will now describe my algorithm. Note that the formalization of the real world problem and the algorithm to solve the problem are heavily linked. This is because the formalization is relatively simple, but we need to coax some modifications on traditional search in order to make this problem computationally feasible to solve.

2.1 Formalization

Initially, this problem seemed well suited to a Markov Decision Problem. In order to do this we must define six components:

- **State** Position, velocity for each agent in the map. Current time. We will also have static parts of the state, which will be the map (boundaries and static obstacles such as walls) and the goal locations for each agent (or groups of agents), which will be represented as rectangles.

- **Actions**(s) New velocity and direction for each agent.
- **T**(s, a) Probability of taking an action is based entirely on sum of rewards for each agent.
- **Reward**(s, a) Distance from destination (perhaps based on a shortest path (taking into account congestion) to goal), closeness of neighboring agents, directions and velocities for each agent etc... This reward function is very flexible, since it is what will most strongly affect the behavior of the agents.
- **Start state** Start state is defined by user, or randomly.
- **isTerminal**(s) Every agent should be inside the rectangle representing the goal location for that particular agent.
- **TerminalUtility**(s) Expend minimal energy.

This formalization is sound. However, there are problems with this formalization that require us to heavily modify our algorithm for solving the problem. Firstly, euclidean space is continuous, so we would have to discretize the space, probably into a grid. We would have to choose how finely to discretize the space. If the grid is too course then we suffer from potentially sub-optimal and unrealistic human motion, but if the grid is too fine then we suffer from unfeasible computing requirements.

Therefore, I propose a different way to solve the problem. We will use random sampling, as well as physically and biologically based methods to solve this search problem. The algorithm below relies heavily on the work in [Guy et al. 2010] and [Kluckhohn 1950], for the idea of using **PLE** (defined below), and [Roussel et al. 1998] for the velocity space optimization method for dealing with agent collisions.

2.2 Algorithm

My algorithm consists of two stages.

- A pre-processing stage. We construct a graph over which agents can use to guide them to their goals.
- Integration stage. Repeatedly, until all agents reach their goal: Compute set of velocities that lead to collisions, and rule them out. Then compute the best next velocity (speed and direction) to travel in, based on the shortest path in the graph (according to energy expended, which we define in detail below). Step the positions and velocities.

I will now walk through the steps in my algorithm.

1. Select a random set of waypoints from the arena. Use stratified sampling, but make sure that no samples are placed in areas of the arena that are off limits to agents (this can be done with a simple point/polygon intersection test. This allows us to model obstructions such as walls.
2. Connect the points into a graph G, where a waypoint v_i corresponds to a vertex and an edge (v_i, v_j, w_{ij}) is the path between v_i and v_j, with weight w_{ij} (weight is an estimate of the time to traverse that edge). We can create the edges using a 2D meshing algorithm, such as Delauney triangulation. This has the nice property that only $O(|V|)$ edges are created, where $|V|$ is the number of waypoints. In this triangulation we must make sure to remove edges that intersect an obstacle.
3. While G is not fully connected remove points from the graph and replace them in a different location. Note that this imposes the restriction that our arenas must also be connected. This could be fixed with a more arena representation.

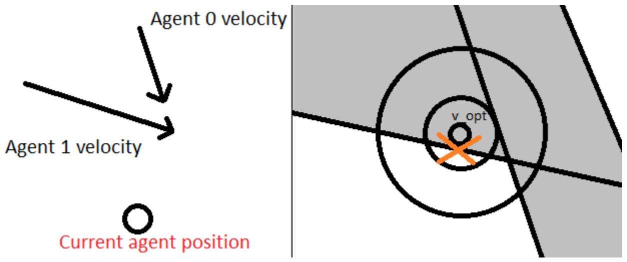


Figure 1 World space and Velocity spaces of the velocities a particular agent can choose. v_opt represents the unconstrained local optimal velocity from our A* and PLE model. The red X marks the constrained optimal velocity for our agent. This is a reasonably simple optimization problem. The gray sections of the velocity space graph represent velocities that result in a collision.

4. Estimate the crowd density around each vertex. To do this we can construct a KD-tree (similar to a 2d binary search tree, except all values are in leaves), which allows fast nearest neighbor lookups. I then find the number of agents per meter squared, n_{ij}. The weight of an edge becomes

$$W(v_{ij}, v_{i',j'}) = \frac{n_{ij}v_{ij} + n_{i'j'}v_{i'j'}}{2}$$

   This serves as a reasonable approximation for the amount of time, or energy it will cost to traverse an edge.
5. Perform A* for each agent on our graph, from the nearest waypoint in between the agent and the goal to get a shortest path to that goal. At this point it is acceptable to assume that agents are allowed to collide with one another.
6. We must now fix potential collisions between agents by computing new positions and velocities for all agents. First we must rule out collisions.

   I use a method proposed by Pradalier *et al* in order to avoid collisions. We project all velocities into a velocity space, with velocity x on one axis and velocity y on the other. It then becomes reasonably easy to check for collisions. We simply cross out the areas of the graph that lead to collisions. This can be done by defining a certain function that makes it infinitely expensive to cross line segments. This is a well known method to solve optimization problems [Boyd and Vandenberghe 2004].

   As an example, see **Figure 1**. We are finding the velocity for an agent A, who could potentially collide with two neighbors (we define a maximum velocity for all agents to make this step more computationally reasonable). So we project into velocity space and compute line segments representing the velocity bounds for each agent. We then restrict the velocities our agent A can take and allow the agent to take the locally optimal velocity decision, based on our PLE model (defined below), combined with a direction determined by our shortest path through G.
7. We compute the magnitude of the velocity (speed) for each agent. I use a simple metric called **Principle of Least Effort** (PLE), first described in 1949 by [Kluckhohn 1950]. It suggests that human agents tend to walk in the direction that minimizes the expected energy that will be expended in moving. Furthermore it suggests that we walk at a pace that minimizes total walking time, but takes into account the extra cost associated with walking fast). This turns out to be a Biologically sound model [Kluckhohn 1950] [Roussel et al. 1998] [Kang et al. 2004].

Number of agents	Boids energy	Crowd simulation energy
10	680.3	510.7
50	919.3	802.1
100	1801	1001.0
500	1701.8	1311.3
1000	1892.2	1912.1

Figure 3 Energy expended per agent with respect to the number of agents, in Joules per agent meter per second.
The figures indicate that my model in almost all cases outperforms that of the Boids simulation. It is not entirely clear as to why in the case of 1000 agents, my model does not perform as well as Boids. It is possible that the close proximity of all the agents means that the velocity space is too constrained for energy efficient models to work well.

8. Step a small time step and integrate the velocities and positions. This is very similar to typical methods for solving differential equations in physically based simulations. I used Euler integration, but larger time steps could be taken by using an implicit integration technique. Note that increasing the timestep too much may lead to collisions or instability, especially when using an explicit method such as Euler.

9. Repeat steps 4-8 until all agents reach their goals.

3 Results

Getting the algorithm to run was a painful task, since my code originally suffered from major integration problems. It turns out that Euler integration is incredibly inefficient and very small time steps must be taken in order to avoid numerical instability.

My approach successfully generated some emergent behavior of crowds, which was good. My algorithm did beat Boids simulation for generating paths for the agents that minimized the average energy expenditure for agents in most of my tests.

One interesting observation was that crowd simulation successfully replicated the well known observation that the average vacating time of an arena by a large group as speed increases is convex. It starts off, as one would expect, with a strolling crowd (¡ 1 m/s evacuates very slowly). However, as speeds increase and go past the average unconstrained speed of a person (1.3 m/s) [Kang et al. 2004], the evacuation time begins to decrease. This is most likely true owing to the fact that as speeds increase the density of the crowd also tends to increase (since there is not infinite room to expand) and so the agents are less able to rely on their shortest path goals, and become more constrained on the number of possible directions they can travel in.

4 Conclusion

I have shown that my algorithm outperforms that of the boids model in most of the tests that I performed. I have also shown that the algorithm successfully reproduces many crowd behaviors exhibited in real crowds.

The algorithm exhibits nice parallelism, due to the fact that each agent's next velocity can be computed independently of the others. Unfortunately I did not get enough time to experiment with parallelization.

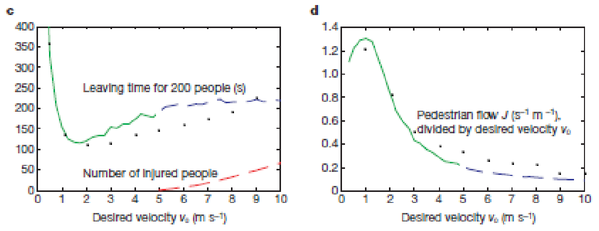


Figure 2 Leaving time for 200 people from an empty room from [Helbing et al. 2000]. My results are the black dots, human behavior is the green curve. My approach, while not being identical to the human results, maintains the same shape of the curves. We can see that my approach does get lower speeds of people leaving a room once the average speed increases above the measured average human walking speed.
Boids simulation was unable to evacuate the room over the period of time I left it running.

4.1 Future Work

There is a lot of space for future work. Firstly, I would like to improve how I generate the original waypoint graph, in order to be more adaptive to complex arenas. At the moment, my algorithm can only really handle relatively simple environments, so it would be nice to scale it to handle many different environments.

Also, I sacrificed much of the 'level of detail' scalability described in other papers. It would have been nice to be able to simulate crowds in high detail in real time in areas that a user might want to concentrate, while still doing a decent job of simulating crowds further away from the attention of the user. It would be interesting to experiment with a hierarchical approach, perhaps using more fluid-like models to simulate very large crowds, while becoming much more like the autonomous car from Assignment 2 from CS221 when there are less than ten agents.

5 Thanks

Thank you very much to all of the faculty who made CS221 an awesome class!

In the interests of not going over the 5 page limit, I have not included code. You are welcome to email if you require a copy.

References

Boyd, S., and Vandenberghe, L. 2004. *Convex Optimization*. Cambridge University Press, New York, NY, USA.

Engber, D., 2006. How did airbus ace its airplane evacuation test?, 03.

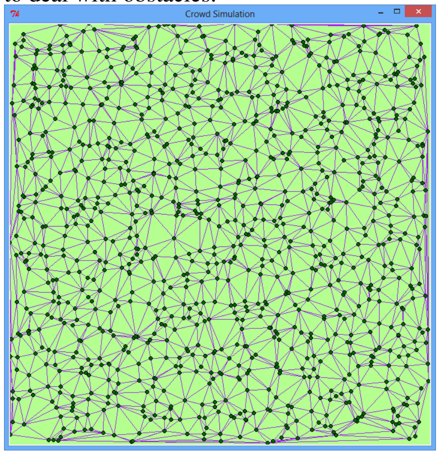
Guy, S. J., Chhugani, J., Curtis, S., Dubey, P., Lin, M., and Manocha, D. 2010. Pledestrians: a least-effort approach to crowd simulation. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, SCA '10,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

Helbing, D., Farkas, I., and Vicsek, T. 2000. Simulating dynamical features of escape panic. *Nature*.

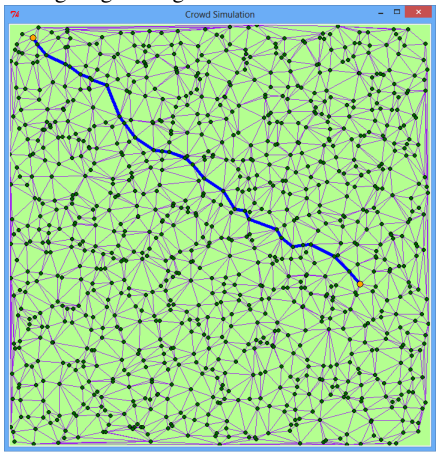
Jund, T., Kraemer, P., and Cazier, D. 2012. A unified structure for crowd simulation. *Comput. Animat. Virtual Worlds 23*, 3-4 (May), 311–320.

SCREENSHOTS

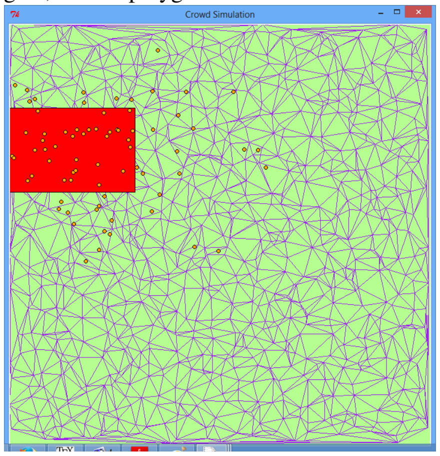
Random waypoint samples, using stratified sampling.
Delaunay triangulation to form the edges, resampling as necessary to deal with obstacles.



A single agent to goal A* search.



100 agents (represented by orange points), making their way to the goal, the red polygon.



KANG, Y.-S., PARK, J.-H., YIM, H.-J., AND SONG, J.-O. 2004. Walking pattern generation for a biped robot using optimized polynomial approximation. In *Humanoid Robots, 2004 4th IEEE/RAS International Conference on*, vol. 2, 707–725.

KLUCKHOHN, C. 1950. Human behavior and the principle of least effort. george kingsley zipf. *American Anthropologist 52*, 2, 268–270.

REYNOLDS, C. W. 1987. Flocks, herds and schools: A distributed behavioral model. *SIGGRAPH Comput. Graph. 21*, 4 (Aug.), 25–34.

ROUSSEL, L., CANUDAS-DE-WIT, C., AND GOSWAMI, A. 1998. Generation of energy optimal complete gait cycles for biped robots. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, vol. 3, 2036–2041 vol.3.