Agent-Based Modelling of Forces in Crowds

Colin M. Henein[1] and Tony White[2]

[1] Institute of Cognitive Science, Carleton University,
1125 Colonel By Drive, Ottawa, Ontario, K1S 5B6, Canada
cmh@ccs.carleton.ca
[2] School of Computer Science, Carleton University,
1125 Colonel By Drive, Ottawa, Ontario, K1S 5B6, Canada
arpwhite@scs.carleton.ca

Abstract. Recent events have highlighted the importance of good models of crowds, however many existing crowd models are either computationally inefficient, or are missing a crucial human behaviour in crowds: local pushing. After discussing some essential aspects of force in crowds, and considering some existing models, we propose an efficient agent-based model of crowd evacuation that incorporates pushing forces and injuries. Basing our model on existing work, we note several problems associated with crowd density in an existing model and recommend solutions. Analysis of our model shows significant effects of force on the crowd, as well as significant effects of crowd density when measuring the number of agents still trapped inside the space after a fixed time.

1. Introduction

Crowds are a part of our everyday lives. While most crowds are safe, some can be dangerous: this year alone over 250 pilgrims were crushed during the Hajj, and over 80 were crushed at the lantern festival in Beijing. Crowd effects are not limited to these exterior settings; for example, architects designing large venues (e.g. arenas and lecture theatres) need to understand crowd exit behaviours. Thus, good models of crowds can be very helpful to many professionals.

From a modelling perspective, crowd events are interesting to social simulation researchers because their associated phenomena are largely emergent in nature. Interesting crowd behaviours with which we are familiar include the formation of rivers of movement through otherwise stationary crowds, spontaneous formation of lanes when pedestrians are moving in opposing directions, and roundabouts when paths cross [1]. Crowds also demonstrate speed and force-related effects, including rainbow-like arching structures as pedestrians jam and clog at exits, bursty exit rates as jostling prevents smooth use of doors, and inability for crowds to pass through each other when speeds (and forces) are high [1]. All these crowd effects are generally observable from an overhead perspective, and we are all aware from our personal experience that individual pedestrians do not plan them.

Force effects are particularly important in modelling crowd behaviours. Although people generally try to move toward goals, force effects can cause them to be pushed away from their desired trajectories and accurate models must reflect this. Also, the

presence of crowd members injured by excessive force can significantly affect the ability of others to move freely. In an evacuation scenario, increased desired walking speed leads to increased forces, and these forces tend to cause additional delays to those trying to exit. Models that do not represent pushing forces therefore cannot directly account for all these additional delays.

The purpose of the present research was to create an efficient, agent-based, individual-centred model of crowd evacuation that incorporates pushing forces and injuries into the model. In the remainder of this section we will consider existing models of crowd behaviour (including some agent-based ones), and discuss their advantages and disadvantages; ultimately we will conclude that one promising model could be improved by the addition of force. Following sections will outline this model, essential aspects of force in crowds, and the details of our implementation of these phenomena. Finally, we will provide an analysis of our results as well as future directions for this research.

1.1 Existing Models of Crowd Behaviour

Many explanations of crowd behaviour have been proposed over the last forty years [review: ref. 1]. Historically, many of these investigations have borrowed physical analysis techniques (e.g. fluid dynamics) that are focussed on aggregate factors. This approach obscures individual-level interactions (e.g. collision avoidance and pushing) which may be important for accurate modelling, and which are of particular interest to researchers seeking to study individual movement and safety.

Recently, agent-based models have been proposed which begin to address this issue by operating at the level of individuals. A frequently cited model of this type is the social forces model, advanced by Helbing and colleagues [2]. A strongly physicalist model, it calculates forces acting on agents to determine movement, with excessive forces leading to agent injuries. The model considers the effect that each agent has upon all the other agents, almost as if the model were a simulation of an *n*-body problem in astrophysics. Physical forces (e.g. friction encountered when brushing past another person, or elastic force due to body compressions) are modelled, as are social forces (desire to change direction to avoid another). One problem with Helbing's model is that of computational complexity. Simulation update is $O(n^2)$ due to the calculation of the effect that each agent (and obstacle) has on all the other agents. This may limit the model's ability to simulate many agents.

A more efficient model may be obtained by avoiding the direct consideration of every agent's effect on all the others. We call these models local-interaction models. Cellular automata models take this approach, as do agent-based models that rely on local information to simulate an unfolding crowd scenario.

Kirchner and colleagues have proposed a model along these lines that emphasizes decentralised processing by independent agents [3]. Agents representing crowd members are distributed on a grid that serves not only as a coordinate system to track agent movement, but also as a data structure storing location-specific data of use to agents. As each agent calculates its next step, references to physical features of interest in the environment (e.g. doors) are replaced by references to a value stored on the grid cell in question (e.g. distance from here to nearest exit).

One deficiency of many local-interaction models is that, unlike Helbing's model, they do not address the question of force applied by agents to other agents. Individuals in Kirchner's crowds, for example, are well behaved as they await their opportunity to exit. This means that the model cannot account for delays in exiting due to pushing or the secondary obstruction effects of injuries. We believe that results can be improved by creating an agent-based local-interaction model incorporating force.

2. Starting with an Existing Model: Kirchner

We have elected to base our model on the Kirchner model [3], as this model is capable of demonstrating many of the effects discussed in the introduction, above.

In this section we will describe the basic model. This will lead us to a discussion of a problem with the model, and our proposed solution. In the next section, we describe how pushing and inter-agent forces are integrated into the model.

2.1 Floor Field Modelling

The modelled space is divided into square grid cells. It makes use of *floor fields* to provide location-specific data to individual agents in the model. Each floor field defines a set of values, one value for each cell. An individual agent has access to the floor field values on its current cell, as well as those on neighbouring cells.

The Kirchner model defines two floor fields: the *static* field and the *dynamic* field. The static field is a gradient that indicates the shortest distance to an exit. Thus, an agent is in a position to know the distance to the closest exit by consulting the static field value on its current cell; by consulting the values in neighbouring cells the agent can determine the *direction* in which to move to reach an exit.

The dynamic field is a measure of agent movement. An agent increments the dynamic field level when moving. By analogy with ant pheromones [4], the dynamic field diffuses and evaporates. By consulting the dynamic field an agent can follow other nearby agents without directly considering the position of any other agent.

By varying the degree to which an agent is sensitive to one field or the other, this model can simulate different pedestrian strategies within the crowd, as well as situations in which pedestrians have less than perfect knowledge of the location of exits. For example, to model herding behaviours in anxious crowds, the agents' sensitivity to the dynamic field can be increased, thereby promoting agent interaction. To model reduced visibility in an unfamiliar environment, agents' sensitivity to both fields can be decreased.

2.2 Update Rules

Kirchner's model proceeds by iterating a set of basic steps. First, a score – representing desirability – is calculated for each cell according to the following formula:

$$Score(i) = \exp(k_d D_i) \times \exp(k_s S_i) \times \xi_i \times \eta_i \quad (1)$$

where *Score(i)* represents the score at cell *i*; D_i is the value of the dynamic field in that cell and S_i is the value of the static field in that cell; k_d and k_s are scaling parameters to the model governing the degree to which an individual is aware of other-agent movement and location of exits, respectively; ξ_i is 0 for forbidden cells (e.g. walls, obstacles) and 1 otherwise; η_i is 0 if an agent is on the cell, and 1 otherwise.

Having determined the scores, each agent then uses them to probabilistically determine the cell that it would like to move onto in the next time step (motion is permitted to the four cardinal neighbours). The agent calculates the probability of moving from cell *i* to a neighbouring cell *j*, by dividing the score for *j* by the sum of the scores of those cells adjacent to *i*:

$$p_{ij} = Score(j)\left(\sum_{a \in N(i)} Score(a)\right)^{-1} \quad (2)$$

where p_{ij} is the desired probability and *N(i)* gives the set of cells that are neighbouring cells to *i*.

When all agents have decided on their desired grid cell, they all move simultaneously and increment the D_i value of their original cell by one. As no two agents may occupy the same space, any conflicts are resolved by allowing a random agent to occupy the desired cell; the other conflicting agents are prevented from moving.

Finally, D_i is diffused and decayed. To diffuse, cells transfer a proportion of their D_i value equally to their neighbouring cells. To decay, cells discard a proportion of their D_i value. The decay and diffusion proportions are input parameters to the model.

Agents moving onto a cell designated as an exit cell are removed from the model, and are deemed to have exited.

2.3 Observed Problems with the Model

We have observed certain problems associated with agent movement preferences that can be traced back to the η term of equation (1). This term aims to prevent two agents from occupying the same cell at the same time by dropping the probability of selecting a currently occupied cell to 0.

The problem with the η term is that it creates a very sparse crowd at exit points. Since movement within the model is simultaneous, one agent should very well be able to leave a cell *c* at the same time as another agent is entering it. However, the η term prevents this by stopping agents from desiring cell *c* if it is occupied, meaning that no agent will try to move into *c* as its current occupant leaves; this guarantees that any occupied cell will be unoccupied for at least one time step when its occupant leaves.

Another problem with the scoring rule is that there is no provision in the model for an agent to remain motionless unless blocked on all sides; agents must take a step if possible, even if it takes them away from their goal. Consider an agent with occupied cells ahead, left and right. According to Kirchner's scoring rule, the agent will have a

100% chance of stepping backwards, because the η term will reduce the probability of selecting the cells ahead, left and right to 0.

Because of these two effects, agents spend a lot of time moving into and out of holes in the crowd. This constant movement and excessive spacing seems at odds with our experience in large crowds, where we generally stay still if we cannot advance towards our goals.

2.4 Resolving η term issues

The model as described contains two provisions aimed at enforcing a rule that no two agents should occupy the same cell. The undesirable effects of the η term can be removed from the model by generalizing the update rule that prevents two agents from moving onto the same cell simultaneously.

One option to resolve the issue, then, would be to simply remove the η term entirely. In some cases, however, it may be desirable to model a crowd jam at lower density. Accordingly, we have redefined the occupied value of the η term as an input parameter to the model, k_n . When a cell is unoccupied then its η is set to 1, and when a cell is occupied then its η set to the value of k_n . This allows us to model situations ranging from closely packed crowds ($k_n = 1$) to very sparse crowds ($k_n = 0$).

3. Modelling the Application of Force

Force is an important aspect of crowd models. Forces in crowds affect the ability of pedestrians to exit in a timely manner. A model without force may not accurately model pedestrian exit behaviours. In this section we add force to Kirchner's model.

3.1 Aspects of Force

Before proceeding to create a model that includes forces it is worth mentioning five essential properties of forces that will guide our design.

First, people in crowds should not push randomly; pushing occurs for a reason. People push in a particular direction when they want to move in that direction and are prevented from doing so. Also, people push when they want to maintain their personal space in a crowd.

Second, forces *propagate through* crowds; some researchers have reported that force can move through a crowd like a shockwave [5]. It is essential to note this fact when modelling force, as it dictates that force applied by individuals in one part of a modelled crowd must propagate through that crowd with certain time characteristics; pushing and force propagation in crowds is not instantaneous.

Third, it should be clear that forces are directed and should be considered to be vectors rather than scalars.

Fourth, from the perspective of a model of force in crowds there are two important effects of force on agents. An agent subjected to extremes of force should become

injured. Injuries should have consequences for the injured agent (immobility, inability to exit) and for agents in the vicinity (injured agents are obstacles to others). When agents are subjected to non-injuring (smaller) forces, these should affect agent movement; a modelled pedestrian who is pushed ought to move in the direction of the force, rather than in some other direction that they might prefer.

Fifth, is location-specificity. If many agents in a crowd are pushing together in a similar direction then the applied force can combine additively to have a large effect on those receiving the force. Often, injuries tend to occur near the front of crowds, as the aggregate force from behind becomes overwhelming. Some reports have concluded that injuries have occurred towards the centre of a crowd, where the force from those pushing off the wall at the front met with force generated by those pushing from the rear [5]. In any case, force and injuries should not be randomly distributed throughout the modelled space, but should instead be produced at physically reasonable positions within the crowd.

3.2 Adding Pushing to Kirchner's Model

When adding force to the model, we have been conscious of the performance advantages of the floor field design. Accordingly, force is represented in the model as a third floor field.

The force field value on each cell represents the force experienced by the agent on that cell in the current time step. In order to represent the directionality of force, we have used a vector field instead of the scalar fields used by Kirchner. We define a vector floor field to hold both a magnitude and direction for each cell in the model, rather than the scalar floor fields that hold only a scalar value for each cell. Where applicable, we use vector addition to update the vector field.

Agent Rule Modifications. The agent applies force in two circumstances. If the agent wishes to move into a cell, but is blocked (another agent has moved there first, or an occupying agent is present) then the agent will push the occupant of the desired cell; in other words, the agent will vector-add its pushing force (parameter k_{push}) into the force field value on the desired cell, in the direction of desired travel.

In the second circumstance of force application, the agent applies force to neighbouring agents in order to maintain its space in the crowd; one quarter of the pushing force ($k_{resist} = 0.25\ k_{push}$) is vector-added into each adjacent occupied cell in the direction of that cell.

The agents follow Kirchner's movement rules, with the following exception: If the force experienced by an agent (i.e. the value of the force floor field at the agent's location) exceeds the force ($f_{\text{divert}} = k_{push} + k_{resist}$) that it can apply to another agent, then the normal score-based probabilistic cell selection described above is bypassed. Instead, the agent is forced to select the cell in the direction of the force. Thereafter, normal movement rules apply (only one agent can actually move onto that cell, etc).

Force Update Rules. The force field is an active field; like the dynamic floor field that tracks agent movement, the force field propagates through the model according to

certain rules. The basic force rule is that as a part of each time step, the entire force on a particular cell moves on to the next cell (in the direction of the force vector). Thus, force is propagated through the model over time. Forces that collide on a cell are vector added together. Force cannot be propagated through empty space; if force becomes stranded on an empty space its magnitude is reduced to zero. Likewise, walls and obstacles absorb force, and do not re-transmit it.

Injury behaviours. Agents become injured if the sum of forces acting on them reaches a threshold level ($f_{injuring}$), which is an input parameter to the model. The scalar sum of incoming forces is used, rather than the vector sum for this purpose. Once injured, an agent ceases movement; its cell is designated as an obstacle cell (like walls).

4. Results

The model was implemented in NetLogo 2.0 [6] (on Mac OS X 10.3.2, Java 1.4.2).

The simulated floor space was 31 × 31 cells in size, not including the wall cells surrounding the floor space. The results presented here are for one exit cell situated in the middle of one wall, with no other obstacles (figure 1a).

Two hundred individuals were assigned random starting cells within the space. Initially the force field had a magnitude of 0 on every cell. The static floor field was initialized according to the linear distance between the centre of each cell and the centre of the exit cell. For the purposes of measuring the effects of force on the model, k_s was set to 10 and k_d was set to 0. The agent's pushing force, k_{push}, was set to 1, and the agent's injury threshold, $f_{injuring}$, was set to 23.

We report performance of the model in terms of the number of agents that remain in the space after a fixed number of iterations. (Although it may seem more natural to report the number of iterations required for all agents to exit, this measure is problematic because if an injury occurs such that an agent blocks the exit it prevents the remaining agents from ever exiting.)

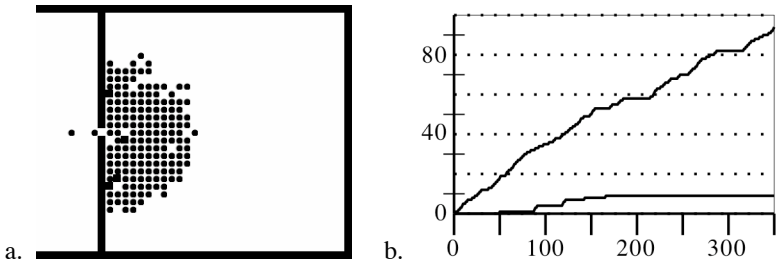

Fig. 1. (a) Snapshot of representative model execution. Agents (*circles*) clustered around door (*white cell in black walls*) in typical arch formation. Injured agents (*black squares*) are present due to previous pushing. (b) Number exited (*upper line*) and injured (*lower line*) with time

4.1 Effect of varying k_n

To investigate the effect of introducing the k_n parameter we measured our model running with Kirchner's rules as a base case ($k_n = 0$). We then measured the result of setting the k_n parameter to values of 0.5 and 1.0.

We observed that increasing k_n increased crowd densities at exits. A sparse crowd was observed when k_n is 0. Arching structures with few empty cells are obtained when k_n is 1. Exits from Kirchner's model occur at quite a regular rate. Increasing k_n to 0.5 causes this rate to increase, remaining regular.

Table 1. Agent escape statistics with Kirchner model rules (means and standard deviations of 10 runs). All differences are significant with a two-tailed Student t-test ($p < 0.0001$) except *

k_n	# remaining after 350
0.0	55.1 (3.7) *
0.5	28.7 (5.7)
1.0	57.7 (4.3) *

4.2 Effect of adding force rules

We measured the effect on agent escape when force was added to the model at the same values of k_n that were tested above.

We observed that forces did propagate through the crowd, gaining strength as multiple agents joined in the pushing action. Forces tend to propagate both into the crowd (towards the vertical centre and towards the door) and out of the crowd (from the vertical centre out) as agents push in the direction they want to move, and simultaneously resist those around them.

Injuries in the model tend not to occur right at the beginning or end of the simulation, but rather begin when a significant crowd density has developed near an exit, and end when enough agents have exited to relieve some pressure. In our trials, injuries tended to occur between time steps 50 and 150 (e.g. figure 1b). Some injuries occurred in groups of 2-4 nearby cells, while other injuries were isolated. The majority of injuries occurred in the half-crowd closest to the door, and tended to be distributed roughly along an arch that passed near the centre of the crowd (at the time the injury occurred). Agents near the wall with the door, but towards the edges of the arch structure were also frequently affected by injuries. We also observed a more bursty exit rate than in trials without force.

Table 2. Agent escapes and injuries with revised model (means and standard deviations of 10 runs). All differences significant with a two-tailed Student t-test ($p < 0.05$)

k_n	# remaining after 350	# injured after 350
0	66.4 (4.7)	0.0 (0.0)
0.5	80.9 (11.1)	4.7 (2.1)
1.0	105.4 (30.8)	7.1 (2.0)

5. Discussion

Revision of Kirchner's cell-selection (scoring) function by introducing a value of $k_n = 0.5$ significantly decreased the number of agents remaining after a fixed time (table 1) compared to other values of k_n. This is because the revised cell-selection rule no longer requires that occupied cells spend a time step unoccupied. Since this rule applies to the exit cell as well, the original model's maximum exit rate was one agent per two time steps, while the new rule allows for a maximum exit rate of one agent per time step. In addition, the new rule increases crowd density, placing more agents in the vicinity of the door, and thereby in a position to readily exit when possible.

At $k_n = 1.0$ agents completely ignore the occupancy of a cell when rating its desirability. This may be a poor strategy in dense crowds as the occupying agent may not be able to move off the desired cell. When k_n is 0.5 agents will select occupied cells but still prefer unoccupied ones – a better trade-off according to table 1.

The addition of force to the model (table 2) results in a significant increase in agents stranded after 350 time steps at all values of k_n when compared with table 1 (two-tailed Student t test, $p < 0.0001$). Hence, the simulation bears out the fact that pushing (paradoxically) increases exit times [1]. It should be noted that the standard deviations increase noticeably when force is added and injuries occur. This is because the number and distribution of injuries play a large role in determining the effect of force on the model. In some cases injuries in front of the door block large numbers of agents, while injuries that occur near the periphery have minimal effect.

Another effect of adding force to the model is that agents tend to exit in bursts instead of at a fairly regular rate. The gap between bursts occurs because an agent *a* standing in front of the exit cell is unable to move towards that cell; this is due to other agents lined up to the left and right of *a* pushing *a* in a different direction, overriding *a*'s desire to step onto the exit cell. Eventually the forces acting on *a* may be disrupted, or overcome by agents behind *a* pushing toward the door, and *a* will exit. This creates a hole that can be capitalised upon by other agents until forces build up to once again jam the door. This bursty exit pattern has been reported as a hallmark of the arches that form at exit points [1].

Bursty exit characteristics may be precluded if injuries occur near, but asymmetric to the door. In these cases, the bursting behaviours are extinguished because the cell with the injury acts as a force break, taking up the force and helping to isolate agents nearer to the door from the paralysing force scenario just described. Force relief by obstacles is a known phenomenon in crowd research [1]. Its effectiveness in regulating bursty output in this model is an effect that was not programmed into the rules; rather it emerges from the effects of force on agents and obstacles. Its emergence in the model is further evidence that our force paradigm is a promising one.

Because crowd density increases with increasing k_n, and crowd density should lead to more pushing and therefore delay and injury, we can further test our model by predicting that delay and injury should increase with k_n when forces are applied by agents. In fact, this prediction is borne out (table 2). Crowd density directly affects the number of injuries because fewer empty cells within the crowd mean more effective force propagation. The increase in density is accompanied by an increase in the number of agents left inside after 350 time steps, both because the number of injuries increases, and because greater force build up results in longer intervals between bursts

of exiting agents. Note that when k_n is 0 no injuries were obtained because spacing in the crowd prevented the transmission and build-up of force.

The principle of location-specificity described above is observed in the model. Generally consistent injury positions re-occur across multiple simulation tests. It is a consequence of the generally semi-circular arching structure that there are more agents pushing on the chord parallel and adjacent to the wall, than on the chord parallel to the wall but near the back of the crowd. The combined pushing power of the larger group of agents along the wall, mostly directed parallel to the wall towards the door, means that injuries tend to occur near this wall. Forces tend to concentrate near the door (result of agents pushing towards the door) and towards the edges of the crowd (result of agents pushing back to maintain space in the crowd).

6. Conclusion

We have described an agent-based model of crowd dynamics based upon local interactions with floor fields. After the addition of a force floor field, the model continued to demonstrate the characteristic arching structure that we expect to see in crowd evacuation simulations. Moreover, our new model is capable of answering quantitative and qualitative predictions related to crowd injuries and egress dynamics.

Problems with low crowd densities in Kirchner's model were addressed successfully by turning his occupancy-exclusion parameter η into a more flexible density control: k_n.

Considerable work remains in the development of this model. In particular, the model needs to be related to real world data, and calibrated so that units of force and distance in the model can be related to real units of force and distance, and so force capability and thresholds are related to human capacities. Finally, a quantitative analysis of the bursting exit behaviours, and role of the k_n parameter in regulating crowd density, needs to be undertaken.

References

1. Helbing, D., Farkas, I.J., Molnár, P., Vicsek, T.: Simulation of pedestrian crowds in normal and evacuation situations. In: Schreckenberg, M., Sharma, S.D. (eds.): Pedestrian and evacuation dynamics. Springer-Verlag, New York (2002) 21-58
2. Helbing, D., Farkas, I., Vicsek, T.: Simulating dynamical features of escape panic. Nature v. 407, 28 September 2000. 487-490
3. Kirchner, A., Schadschneider, A.: Simulation of evacuation processes using a bionics-inspired cellular automaton model for pedestrian dynamics. Physica A 312 (2002) 260-276
4. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm intelligence: From natural to artificial systems. Oxford university press, New York (1999)
5. Fruin, J.: The causes and prevention of crowd disasters. In: Smith, R.A., Dickie, J.F. (eds): Engineering for crowd safety. Elsevier, New York (1993)
6. Wilensky, U.: NetLogo. Center for connected learning and computer-based modeling, Northwestern University, Evanston IL. 1999. http://ccl.northwestern.edu/netlogo