Simulation of City Evacuation Coupled to Flood Dynamics

A.S. Mordvintsev, V.V. Krzhizhanovskaya, M.H. Lees, and P.M.A. Sloot

Abstract Crowd modeling is one of the key components of risk analysis and evacuation planning in emergency situations. This paper presents a simulation environment for experimenting with different city evacuation scenarios. The simulation couples a flood model with a crowd escape model. The developed agent-based crowd model mimics the behavior of pedestrians escaping from dangerous regions towards safe areas. The system is evaluated through a series of experiments, modeling the flooding of an area in St. Petersburg, Russia.

Keywords Crowd dynamics • Evacuation • Urban flood • Agent based simulation • Decision support

A.S. Mordvintsev (✉)
National Research University ITMO, St. Petersburg, Russia
e-mail: A.S.Mordvintsev@gmail.co

V.V. Krzhizhanovskaya
National Research University ITMO, St. Petersburg, Russia

University of Amsterdam, Amsterdam, Netherlands
e-mail: V.Krzhizhanovskaya@uva.nl

M.H. Lees
Nanyang Technological University, Singapore, Singapore
e-mail: mhlees@ntu.edu.sg

P.M.A. Sloot
National Research University ITMO, St. Petersburg, Russia

University of Amsterdam, Amsterdam, Netherlands

Nanyang Technological University, Singapore, Singapore
e-mail: P.M.A.Sloot@uva.nl

U. Weidmann et al. (eds.), *Pedestrian and Evacuation Dynamics 2012*,
DOI 10.1007/978-3-319-02447-9_40,

1 Introduction

Global warming and rising sea levels have seen an increase in the severity and frequency of urban floods worldwide. Governments are now struggling to understand and provision for possible future emergencies that may involve mass evacuation of large cities. This problem is especially true for countries with cities that lie close to sea level, and those cities with large surrounding water mass. St. Petersburg, Russia is one such city. With a large sprawling river system and a central island surrounded by water, the city has historically experienced a number of major floods, many of which have had significant impact [1]. In order to better prepare for such floods it is possible to use simulation as a means of understanding how different flooding scenarios might impact the city and how the flow of water might be controlled to mitigate the overall impact [2–8].

In some cases, particularly with increasing severity of flooding, there is real risk of fatalities in the city population. Understanding how best to evacuate or manage the people of the city should help reduce the likelihood and level of danger in such scenarios. Again, simulation can be used to help analyze these questions. In this context crowd modeling is one of the key components of risk analysis and evacuation planning for emergency situations. However, to fully analyze and prepare the correct evacuation measures it is essential to understand the interacting dynamics of the flood and the human populations. In particular we want to be able to understand for a variety of possible flood scenarios, what the best strategy is for ensuring the safety of the population, and how to adapt the environment in order to increase the survivability of the city populous.

There are several well-known methods to assess the impact of the flood to the urban areas. Some of them consider the economical effects of flooding [9], while others focus on estimation of the number of fatalities [10–12]. Some of these use empirical models which try to predict the number of casualties by some heuristic rules. For example [11] uses numerical flood simulation to estimate the flood characteristics (depth, flow speed and so on) in the modeled area. They apply an empirical formula to estimate the fraction of fatalities (mortality function) for each spatial location. Given the initial population distribution this can provide a useful estimate of the number of loss of life events.

Some authors use agent-based modeling (ABM) to employ more details of population behavior into the simulation. For example [10] uses ABM to model the cycle of daytime routines being interrupted by a flood. The authors consider the case of vehicle evacuation over a road network. The Life Safety Model [12] is another agent-based model of flood evacuation; it uses a sophisticated set of rules to predict loss-of-life events depending of agent's health condition and actual location (building, vehicle, pedestrian).

In this paper we present an egress model which is coupled to the output of a flood simulation. This coupling of models provides a simulation environment with which different city evacuation scenarios can be evaluated. The main difference of this paper to the previous work is the strong orientation towards pedestrian

crowd dynamics. Our model tries to reproduce the effects of junctions and different pedestrian path planning strategies. The simulation couples a flood model with a crowd escape model for a scenario of flooding of a region in St. Petersburg, Russia.

The model consists of two main parts: a hydrodynamic flooding model and a crowd escape model. We use a Flood Simulator [9] adapted by the UrbanFlood project [2] from a DRFSM code developed by HR Wallingford for simulation of floodwater propagation. It receives water flow rates discharged into floodplain areas from breached or overtopped flood defenses and then spreads the water over the floodplain according to the city topography. The primary outputs of the Flood Simulator are water levels and flow velocities in the area of the city. These outputs form the inputs to an agent-based model of egress that mimics the behavior of pedestrians heading from dangerous regions towards safe areas. More explicitly, it uses the output of the flood simulation to trigger the evacuation process, to compute available evacuation paths and to identify those agents trapped in, or killed by, the flood.

The remainder of this paper is organized as follows: in Sect. 2 we give an overview of our modeling approach; Sect. 3 describes the details of flood model interoperation; Sect. 4 explains the crowd behavior model, including the path planning and collision avoidance algorithms; in Sect. 5 we show simulation results for several evacuation scenarios and model parameters; Sect. 6 concludes the paper.

2 Modeling Approach

The novelty of our work is that we essentially couple two simulations: water spreading on the lands due to overtopped defenses or breached dams, and an agent-based simulation of pedestrians trying to avoid the flood and to reach the safe areas. This section briefly describes both of those models and outlines the execution flow of the combined simulation environment.

The basic outline of the simulation execution is shown in Fig. 1. The first stage is to run a flood simulation, which generates output containing details of the flood dynamics in the city. This data is used to instantiate the environment of the agent-based crowd model.

Every agent in the crowd model is characterized by the following set of attributes:

- Spatial position in continuous 2D space
- Behavior state, one of: *idle*, *running*, *safe*, *drowned*
- Desired speed.

Other agent parameters are given in Table 1.

Initially all agents are considered to be in the *idle* state. Currently we use a uniform initial distribution of agents in empty areas of the modeled region.

The crowd model proceeds in a time-stepped fashion and takes the following actions at each time step of the simulation:

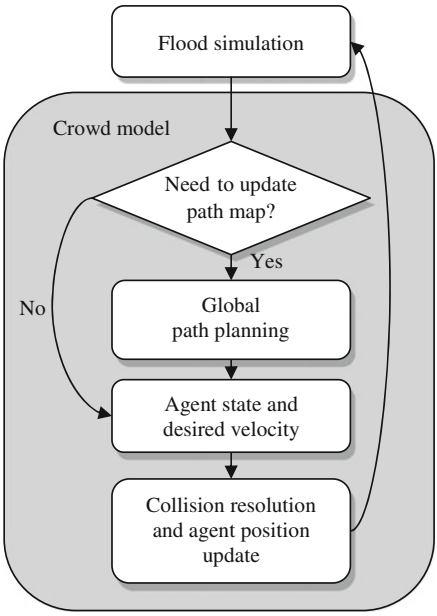

Fig. 1 Outline of flood/evacuation simulation

Table 1 Default model parameters

Model parameter	Value
Number of simulated agents	10,000
Grid size (height map, path map, obstacle map)	782×531
Grid step	9.42 m
Agent preferred velocity (mean, std)	4.0, 0.2 km/h
Agent maximum velocity	8.0 km/h
Probability of a *running* agent to alert the neighbors within a second	0.02
Agent alert neighborhood radius	36 m
Crowd simulation time step	0.5 s
Radius of a circle approximating agent body size	0.5 m
Path map update interval	30 s of model time

- **Flood simulation**. Our model treats the flow simulator as an external component, which provides information about the flood at each time step. Currently the Flood Simulator described in [9] is used, but in principle it is possible to couple the agent-based simulation with any code for flood simulation. The output of the Flood Simulator is converted into a raster flood map, which is a binary image, marking grid sites that are considered to be flooded. Time steps of the flood simulation output may be much longer, than the steps of agent-based crowd model, so we adopt a simple method of interpolation between subsequent generated flood maps. More details about the flood model and associated interpolation are given in Sect. 3.

- **Global path planning**. Next, a global path map to the safety zones is updated. We use a potential function path planning approach [13]. For this we compute a potential function on an obstacle map, combining static obstacles with non-traversable flooded areas. For performance reasons this calculation is performed only in a subset of the iterations. Currently we specify a fixed predefined model time interval to define the frequency of this calculation. We cache the results of each execution of this step for use in subsequent runs of the model (with the same flood and path planning parameters).The use of global path planning rests on the assumption that all agents know the entire city map and the current flood situation. Such an ideal scenario is in principle possible, but would require some technology capable of providing complete real-time information transfer between the agents. A more detailed description of our path planning algorithm and its variants can be found in Sect. 4.1.
- **Agent state and desired velocity update**. Following behaviours are executed:

  (a) *Idle* agents are stationary. If an agent senses the approaching water, its state is changed to *running*. We sample the state of the flood at the agent's current position at a time shifted 5 min into the future. This way we emulate prediction capabilities of an agent. If an agent observes another agent in the *running state* then the observing agent transits to a *running* state too with some predefined probability.

  (b) *Running* agents are heading towards the Safe Heavens. The potential-based navigation model causes agents to simply try to follow the negative gradient of the potential function, computed at the path planning stage.

  (c) Agents that manage to reach the Safe Heavens change their state to *safe*. Agents that happen to be in flooded areas are assumed *drowned*. Agents in either of these states will be removed from the simulation.

  The output of this stage is a desired velocity vector for each agent. For *running* agents this vector points in a direction suggested by the path planning approach and has a magnitude equals to agent's desired speed.

- **Collision resolution and agent position update**. We use the RVO2 library [14] to implement agent-agent and agent-obstacle collision avoidance logic. The library uses the vector obstacle map to compute agent velocity vectors avoiding collisions. The details of scene data preparation and RVO operation are given in the Sect. 4.2. We then use the computed velocities to obtain new positions of agents for the next iteration of the simulation.

3 Flood Simulation

The flood simulation model receives water flow rates discharged into floodplain areas from breached or overtopped flood defenses and then spreads the water over the floodplain according to the city topography. We use a Rapid Flood-Spreading model, described in [9] for floodwater propagation. The model inputs are:

- Raster height map of modeled area,
- Positions and rates of floodwater discharges.

The simulation splits the modeled region into so called *impact zones* (IZ); aggregations of a number of adjacent height map cells. The model iteratively updates the state of the impact zones. Periodically the model stores the amount of discharged water, water level and flux for each IZ affected by spreading flood into a database table. We convert the model data so as to form an output raster map of water levels on grid cells. Any other source of data which could be converted to this form can be used by the simulation. Each grid cell is classified as traversable or not traversable, depending on its flood condition. Currently we simply consider raster map cells with water level greater than zero as flooded and not traversable.

As previously stated, the agent-based pedestrian crowd and flood models (in general) may work on significantly different time-scales. A typical reasonable time step for an agent-based simulation of pedestrian behavior would be in the order of seconds (or sub-seconds). Large-scale flood models on the other hand will usually adopt a time-step on the order of minutes or even tens of minutes. Moreover, in some cases one might want to use historical flooding data, or data acquired from real world (life) observations, so low temporal resolution might be an inherited property of the data used. For these reasons it is desirable to have a method capable of interpolating between coarse flooding time steps. We use the following strategy to do this. Consider two consecutive flooding time steps at t_0 and t_1. Proceed through following steps to interpolate between them:

- Build raster traversability maps for each of them: $f_0(x,y)$ and $f_1(x,y)$, where $f_i(x,y) = 1$, if raster grid cell is not traversable (flooded) and 0 otherwise. We make an assumption that $\forall\, x, y\, f_0 \leq f_1$ i.e. flooded areas don't become traversable again during the simulation. We need to interpolate smoothly between binary functions f_0 and f_1 for values of $t \in [t_0,t_1]$.
- Build two *distance transforms*: $d_0(x,y)$ of function $1 - f_0(x,y)$ and $d_1(x,y)$ of function $f_1(x,y)$. Distance transform computes the distance to nearest zero pixel for each pixel of the input image in terms of given distance function (we use Euclidian norm). We use an OpenCV [19] implementation of an algorithm described in [15] for the construction of the distance transform.
- Compute a *morph coefficient* $w = d_0/d_0 + d_1$ for each pixel in the transition zone, where $f_0 \neq f_1$. Then the interpolated traversability function:

$$f_t(x, y) = \begin{cases} f_0, & |\, f_0 = f_1 \\ 1, & |\, f_0 \neq f_1, w \leq w_t \\ 0, & |\, f_0 \neq f_1, w > w_t \end{cases}$$

where $w_t = (t - t_0)/(t_1 - t_0)$ *is a morphing* threshold for a particular time value (Fig. 2).

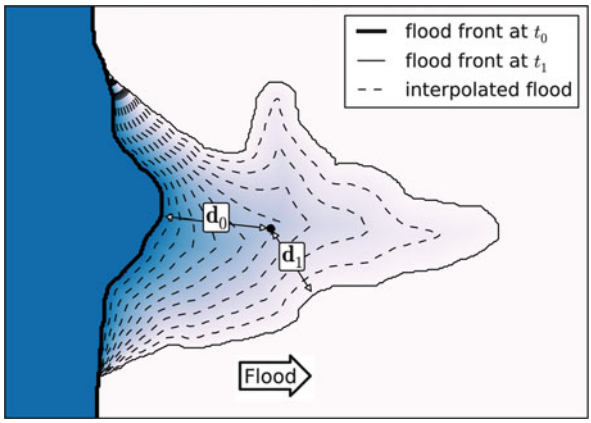

Fig. 2 Flood map spatio-temporal interpolation

4 City Evacuation Model

Our agent-based crowd model mimics the behavior of pedestrians heading from dangerous regions towards the predefined safe areas. Inputs of the model are:

- City topography data, including raster height map (also used for flood modeling), and a vector obstacle map, which contains locations of buildings, fences and other obstacles. Some buildings or map areas are marked as Safe Havens.
- Demographics data, including population distribution of people across the modeled area, the distribution of walking speeds and other agent characteristics.
- Results of hydrodynamic modeling of the flood in the form of a traversability mask for a current time step, as described in Sect. 3.

We use the raster height map as a topography model of the region. This is the same height map that is used by the flood simulation. We classify all grid cells as either obstacles or empty spaces. In a used height map all elevations higher than 3 m belong to buildings, this produces a reasonable obstacle map. For regions with more complex obstacles (high fences and low buildings) a more sophisticated strategy is necessary with tracking steep elevation gradients.

Pedestrian behavior models often separate the logic of agent behavior into several layers [17, 18], where each subsequent layer controls the underlying layer and receives feedback from it. The decisions made by each layer depend of the agent state (memory) and perception. In our model we split the agent motion model into path planning and collision avoidance. Every time step, the path-planning subsystem computes a *desired velocity vector* for each agent. Currently in our model this vector is guaranteed to lead *running* agents to the Safe Heavens. This is a simplified approach that can be replaced by a suboptimal path planning with intermediate destinations. The magnitude of the *desired velocity vector* is equal to the agent's preferred speed. For agents in other states their desired velocity is assumed to be zero. Desired velocity vectors are passed to the collision avoidance subsystem,

which assigns agents new, real velocities based on their desired velocities and current velocities of their neighbor agents. In the following subsections we describe approaches to control the agent's behavior on both layers used in the model.

4.1 Path Planning

We use the Artificial Potential Fields (APF) method for the agents' path-planning. This is a well-known path-planning tool [13], here the basic idea is to define a potential function φ over the modeled space in a way that following the negative gradient of φ leads agents to the destination, while avoiding collisions with static obstacles. There are many methods for constructing such functions discussed in the robotics literature [20]. Many of those papers discuss simple yet efficient techniques like using a superposition of attractive APFs of one or more targets and repulsive APFs from obstacles.

The advantages of such techniques are computational efficiency, implementation simplicity and the ability to produce smooth and safe paths. One of the major drawbacks is the sensitivity to unwanted local minima of the used potential functions. There are some suggested strategies for addressing this issue that try to detect when the agent is stuck in spurious minimum and modify the APF in an attempt to eliminate it (e.g. by varying obstacle repulsive fields or avoiding previously visited locations).

Another way to avoid the local-minima problem is to explicitly construct an APF with minima in destination points only. Several approaches for the construction of such functions (sometimes called *navigation functions* [21]) are commonly used, methods based on *eikonal* [23] or *Laplace* [22] equations are particularly popular. Examples of different navigation functions are shown in Fig. 3.

In our work, the Eikonal equation is used. We solve a partial differential equation that governs the behavior of a potential function φ in spatial domain D:

$$\nabla\varphi(x) = c(x), \quad c(x) > 0, x \in D \backslash T \tag{1}$$

$$\varphi\,(\mathrm{x}) = 0, \quad \mathrm{x} \in \partial T,$$

where T denotes target regions and $c(x)$ is a positive cost function. The cost function is equal to infinity in obstacle regions. In traversable space regions $c(x)$ equals to unit cost (in time or fuel) of passing through a particular point. It is possible to modify an agents' path-planning strategy by varying this cost function. For example, Fig. 3a shows an example of heading to a destination while avoiding moving too close to obstacle boundaries. Figure 3b shows an alternative strategy, where the agent tries to move along the obstacle boundaries. This is achieved by setting the cost function near obstacles to higher or lower value than in the open space.

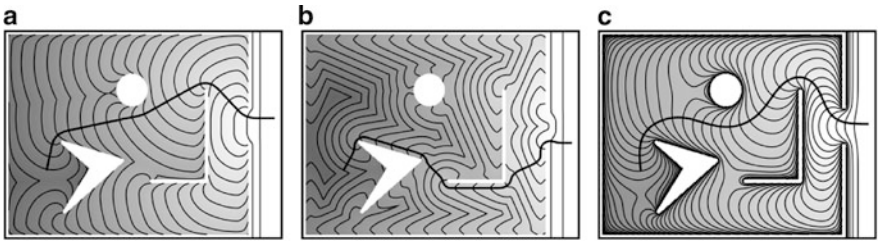

Fig. 3 Potential functions contour lines for: (**a**) eikonal equation with wall avoidance, (**b**) eikonal equation with wall following, (**c**) Laplace equation

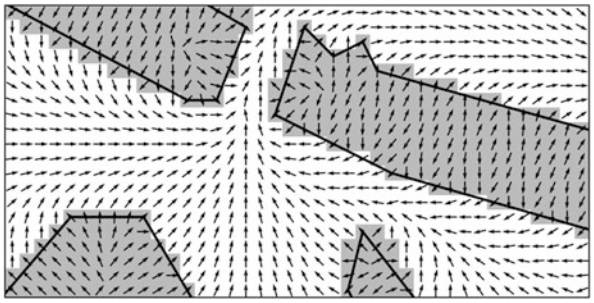

Fig. 4 Part of obstacle map (buildings are shown in *grey*) with vectorized obstacles and navigation map directions

It can be shown [21], that for any point in D following the direction of $-\nabla\varphi$ will lead to a given destination region by a minimal cost path. Absence of unwanted local minima is guaranteed by explicitly setting $\nabla\varphi \neq 0$ for all traversable space points.

We use a numerical solver based on a modified Dijkstra algorithm to solve Eq. 1 on a two-dimensional Cartesian grid. Intuitively it can be described as a model of shock-wave propagation. The algorithm starts from target sites and adds nodes to a visited set one-by-one. Every time step, it takes a node with the smallest estimated wave front arrival time and re-calculates arrival time for the unvisited neighbors. More details can be found in [23]. While being reasonably efficient, this algorithm suffers from poor scalability due to its sequential nature. Parallel methods of solving Eq. 1 are preferable; some methods can even been adapted for GPU computing [24].

Figure 4 shows a close-up picture of an obstacle map. Black arrows indicate the direction of the steepest decrease of the potential function. Grey cells indicate grid sites marked as obstacles. It can be seen on the figure that the navigation function is defined even for obstacle cells. This is done because we approximate raster obstacle contours with simplified polygons used by the collision avoidance model. Some cells, marked as obstacles may fall of this approximation. We use the Douglas–Peucker algorithm [16], which guaranties that the resulting polygonal approximation lies within a certain given threshold from the initial contour. Very low thresholds generate contours that follow the raster defects of the obstacle map.

To avoid this we use a higher threshold value (one pixel in this example), which gives a more realistic contours but causes some boundary obstacle pixels to fall off the polygonal obstacle boundary approximation. If an agent gets to such point it still needs a defined navigation direction to continue its motion. However our path planning algorithm stops on raster obstacle boundaries and does not provide any meaningful direction inside them. To avoid confusion we provide some reasonable moving direction for grid sites marked as obstacles. To do this, we run two path planning passes: one for empty spaces with destinations in the Safe Heavens and the other one for obstacles with destination in empty spaces. The combined navigation map is shown in Fig. 4.

4.2 Collision Avoidance

We use the collision method implemented in the Optimal Reciprocal Collision Avoidance (ORCA) method [14] for local collision avoidance between agents. The implementation uses RVO2 library. The library analyses each agent's neighborhood and for each neighboring agent or obstacle the library computes a subset of allowed velocities, which avoid collision with this particular agent or obstacle in a given time interval. If the intersection of these subsets is non-empty, the library uses 2D convex linear programming to find allowed velocity vector closest to the agent's preferred velocity. Otherwise it tries to find a velocity that minimizes the overlap of the agents. In this case the new agent's velocity doesn't depend on its preferred velocity and it "goes with the flow". In our simulations interpenetrations between agents were reasonably small even in crowded conditions.

The standard RVO2 library uses polygonal representations for static obstacles. We use the vectorization procedure described earlier. Note that we vectorize static obstacles only. Avoiding flooded regions is achieved by path planning only, not using collision avoidance.

5 Results

In this section we describe the results of extensive runs of our model with varying sets of parameters. An example of simulation results is shown in Fig. 5. At the beginning of the simulation all agents are IDLE and evenly distributed in the streets. Then some agents start evacuating as the alert zone (shown in grey) spreads from the left part of the scene.

Agents move towards the safe zones and alert other agents of their way. This process spawns an avalanche of evacuating agents. Our default simulation settings are based on trying to provide reasonable values for all parameters. Table 1 gives a list of used model parameters:

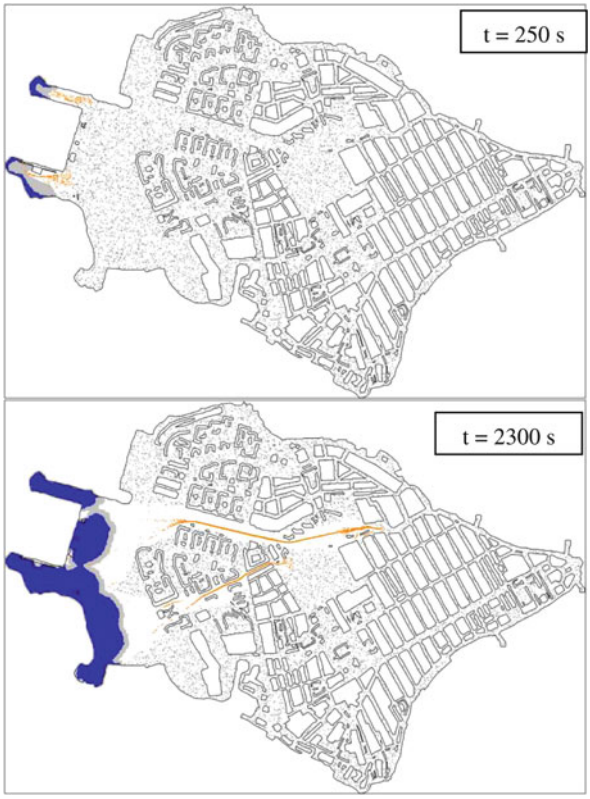

Fig. 5 Two time steps of the evacuation simulation. Evacuation is triggered as a reaction to the flooding of Vasilievsky Island in St. Petersburg. *Dark regions* show the flooded zones. *Gray zones* are "alert zones" where agents see the flood and start evacuating. *Orange points* are the running agents. *Grey points* are agents unaware of the approaching flood

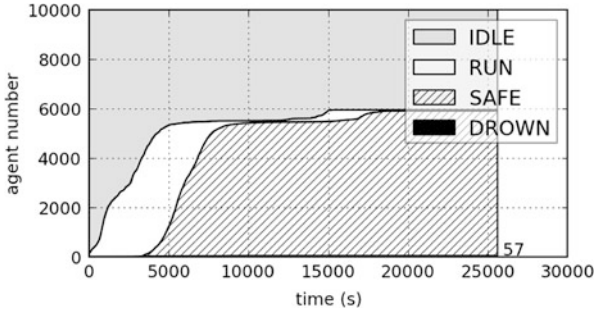

Fig. 6 Agents' states evolution in time for default model parameters

Our default settings allowed almost all agents to escape the flood. The evolution of agents' states over time is shown in Fig. 6.

The number of casualties equals 57 for this simulation. Interestingly, most of these casualties are the result of agents being cut off from the evacuation routes due to the rapid flood spreading.

To study the influence of simulation parameters on the simulation results, we conducted a series of model runs with varying mean agent desired velocities and

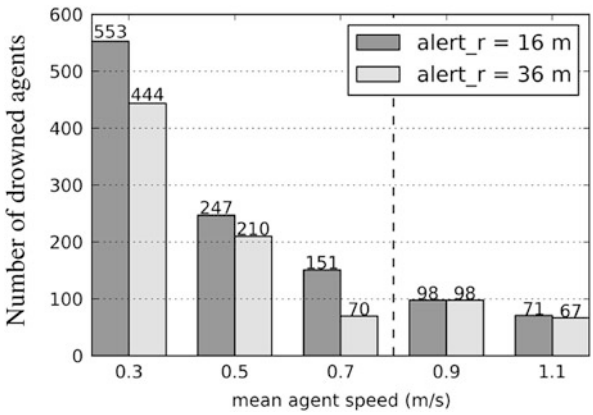

Fig. 7 Number of drowned agents for different agent parameters

alert radices. Resulting numbers of casualties for different simulation parameters are shown in Fig. 7.

As expected the number of drowned agents decreases as mean velocity increases. It can be observed that for small agent velocities (or a rapidly approaching flood) the ability of agents to spread information about the flood can play an important role in decreasing the number of casualties. For example, for a mean velocity equal to 0.7 m/s, increasing the alert radius leads to more than a twofold decrease in casualties.

The agent alerting capability is far less important when the velocity of agents is sufficient to avoid the approaching flood water, even without preliminary notification. These cases are shown to the right of the dashed line in Fig. 7.

The current distance-based alerting approach is quite sensitive to the initial distribution of agents. The experiments were conducted by initializing agents using a uniform distribution in empty areas of the modeled region. Results may become very different for different agent density (this currently depends on the number of agents) or distribution structure (e.g., sparse set of dense clusters).

5.1 Junction Effects

Our simulation region has enough room to provide uncongested flow of agents evacuating from the flooded areas to the safe zones. In order to test the behavior of our model in crowded scenarios we reduced the grid step parameter of the simulation from 9.5 to 2.0 m, leading to a more than 22 times region area decrease. This led to a rapid spread of the flood alert, since the agent alert radius was not modified. As Fig. 8 shows, all the agents quickly run towards the exits, causing congestion in narrow passages.

Road junctions limit the rate of agents leaving the modeled area. The line separating the IDLE and RUNNING zones is much steeper than the RUN-SAFE

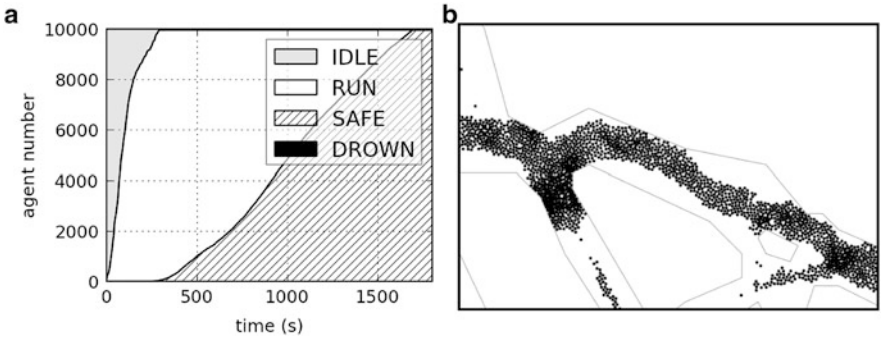

Fig. 8 (**a**) Agents' states evolution for a crowded scenario. (**b**) Crowd flow junction

line in Fig. 8a. Compare this to Fig. 6, where the number of evacuated agents almost follows the number of alerted agents up to the delay introduced by agents traveling time.

Figure 8b shows an example of a close up of such road junction. While the RVO collision avoidance model behaves reasonably well in low and medium crowd density conditions, it allows some agent interpenetration at high density condition. Such zones of crowd "thickening" might be the signs of dangerous regions, where high pressure levels may cause injures. However, the geometrical nature of the RVO model does notconsider such agent properties as mass, force, friction, etc. Therefore it does not provide a quantative way to measure the level of pressure inside a crowd.

5.2 Performance

Our system handles scenarios containing up to 10,000 evacuating agents at 77 computational time steps per second. With a time step of 0.5 s, we simulate evacuation 38 times faster than real time. We run path map update procedures every 30 s of model time. Calculations of the path map took about 22 % of the execution time. Note that we cache computed path maps, so subsequent runs of our model with the same environment and path planning parameters ran about 20 % faster. The remaining execution time is dominated by sensing (querying agent neighbors) and avoiding collisions. Performance was measured on an Intel Core i7 2.93GHz quad-core CPU.

6 Conclusion and Future Plans

In this paper we have presented a model of city-scale evacuation which is coupled to a rapid flood dynamics model. Through experimentation we have shown that the proposed model is capable of producing visually adequate simulations of

crowd evacuation. Beyond this, our experiments offer a number of interesting insights which we plan to investigate in the future. Below we summarize these key observations.

While an increase in agent velocities allows more agents to escape the flood, there is still a residual number of casualties (Fig. 7). Some of these may be avoided by improving the agent path planning strategy, both in the model and reality. For example, avoiding zones that are expected to be flooded soon, via some form of forecast, might be one way to improve the survival rate. Another important source of casualties is groups of agents trapped in isolated areas due to initial rapid spreading of the flood. These observations emphasize the importance of careful evacuation path planning, especially when considering the possible scenarios of flood development. One important factor that was shown to influence the results of evacuation is the ability of agents to sense the approaching flood. Related to this observation, we found the ability of agents to spread the flood alert further around the island can improve survival in some cases. The importance of this ability decreases to being almost insignificant in cases where agents are able to run fast enough to escape the approaching flood, hence reducing the need for prior alert.

In future we plan to make a few important extensions to our model, such as individual randomized graph based path planning, more sophisticated information transfer between agents, incorporation of social relations and walking-in-water models. One final, and most important, open question is the model validation. We plan to investigate the possibility of conducting an experiment by reproducing some historical events (e.g. [1]).

Acknowledgement This work is supported by the *Leading Scientist Program* of the Russian Federation, contracts 11.G34.31.0019 and 13.G25.31.0029; by the EU FP7 project *UrbanFlood*, grant N 248767; and by the *BiGGrid* project BG-020-10 # 2010/01550/NCF with financial support from the Netherlands Organisation for Scientific Research NWO.

References

1. K.S Pomeranets. Three centuries of floods in Petersburg (in Russian: Три века петербургских наводнений). St. Petersburg: Iskusstvo-SPB, 2005.
2. V.V. Krzhizhanovskaya et al. *Flood early warning system: design, implementation and computational modules*. Procedia Computer Science, V. 4, pp. 106–115, 2011. http://dx.doi.org/10.1016/j.procs.2011.04.012
3. A.L. Pyayt et al. *Artificial Intelligence and Finite Element Modelling for Monitoring Flood Defence Structures*. Proc. 2011 IEEE Workshop on Environmental, Energy, and Structural Monitoring Systems. Milan, Italy, September 2011, pp. 1–7 http://dx.doi.org/10.1109/EESMS.2011.6067047
4. B. Pengel, G.S. Shirshov, V.V. Krzhizhanovskaya, N.B. Melnikova, A.R. Koelewijn, A.L. Pyayt, I.I. Mokhov. *Flood Early Warning System: Sensors and Internet*. IAHS Red Book series, 2012 (in print)
5. N.B. Melnikova, G.S. Shirshov, V.V. Krzhizhanovskaya. *Virtual Dike: multiscale simulation of dike stability*. Procedia Computer Science, V. 4, pp. 791–800, 2011. http://dx.doi.org/10.1016/j.procs.2011.04.084

6. N.B. Melnikova et al. *Virtual Dike and Flood Simulator: Parallel distributed computing for flood early warning systems.* Proc. Int'l Conference on Parallel Computational Technologies (PAVT-2011). Publ. Centre of the South Ural State University, Chelyabinsk, pp. 365–373. http://goo.gl/MPLfk
7. A.L. Pyayt et al. *Machine Learning Methods for Environmental Monitoring and Flood Protection*. World Academy of Science, Engineering and Technology, Issue 78, pp. 118–124, June 2011. http://www.waset.org/journals/waset/v78/v78-23.pdf
8. B. Gouldby et al. *Multiscale modelling in real-time flood forecasting systems: From sand grain to dike failure and inundation*. Procedia Computer Science, V. 1, p. 809. http://dx.doi.org/10.1016/j.procs.2010.04.087
9. B. Gouldby et al. A methodology for regional-scale flood risk assessment. Proceedings of the Institution of Civil Engineers, 2008.
10. Dawson R.J., Peppe R. & Wang M.A. Agent-based model for risk-based flood incident management. Nat Hazards 2011, 59, 167–189.
11. M. Di Mauro, K.M. de Bruijn. Application and validation of mortality functions to assess the consequences of flooding to people. Journal of Flood Risk Management. Volume 5, Issue 2, pages 92–110, June 2012.
12. M. Di Mauro and D. Lumbroso. Hydrodynamic and loss of life modelling for the 1953 Canvey Island flood, in the proceedings of the International Conference on FloodRisk, 30 September to 2 October 2008, Oxford, UK
13. Khatib, O. Real-time obstacle avoidance for manipulators and mobile robots. Proceedings. 1985 IEEE International Conference on Robotics and Automation,1985.
14. Jur van den Berg, Stephen J. Guy, Ming C. Lin, and Dinesh Manocha, "Reciprocal n-body collision avoidance," in CédricPradalier, Roland Siegwart, and Gerhard Hirzinger (eds.), Robotics Research: The 14th International Symposium ISRR, Springer Tracts in Advanced Robotics, vol. 70, Springer-Verlag, Berlin, Germany, May 7, 2011, pp. 3–19.
15. P. Felzenszwalb, D. Huttenlocher. Distance Transforms of Sampled Functions. Computing and Information Science Technical Reports, Cornell University, 2004.
16. D. Douglas, T. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. Cartographica: The International Journal for Geographic Information and Geovisualization, V. 10, N 2. (1 October 1973), pp. 112–122.
17. C. Reynolds. Steering Behaviors for Autonomous Characters. 1999.
18. Suiping Zhou, Dan Chen, Wentong Cai, Linbo Luo, Malcolm Yoke-Hean Low, Feng Tian, Victor Su-Han Tay, Darren Wee Sze Ong, Benjamin D. Hamilton: Crowd modeling and simulation technologies. ACM Trans. Model. Comput. Simul. 20(4): 20 (2010).
19. G. Bradski. The OpenCV Library. Dr. Dobb's Journal of Software Tools, 2000.
20. C.W. Warren. Global path planning using artificial potential fields. IEEE International Conference on Robotics and Automation, 1989.
21. S. LaValle. Planning Algorithms (First ed.), Cambridge University Press, 2006.
22. C. I. Connolly , J. B. Burns , R. Weiss. Path Planning Using Laplace's Equation. In Proceedings of the IEEE International Conference on Robotics and Automation, 1990.
23. L. C. Polymenakos, D. P. Bertsekas, J. N. Tsitsiklis. Implementation of efficient algorithms for globally optimal trajectories. IEEE Transactions on Automatic Control In Automatic Control, Vol. 43, No. 2., pp. 278–283, 1998.
24. Won-ki Jeong, Ross, T. Whitaker. A fast eikonal equation solver for parallel systems. In SIAM conference on Computational Science and Engineering, 2007.