MAKKSim: MAS-Based Crowd Simulations for Designer's Decision Support

Luca Crociani, Lorenza Manenti, and Giuseppe Vizzari

CSAI - Complex Systems & Artificial Intelligence Research Center
University of Milano-Bicocca
Viale Sarca 336, 20126, Milano, Italy
luca.crociani@unimib.it,{manenti,vizzari}@disco.unimib.it

Abstract. This paper presents MAKKSim, a pedestrian dynamics simulator based on a computational discrete model in which pedestrians are represented as utility-based agents. The computational model and the system architecture are discussed, focusing on the development of the tool and on its application in a real-word case study, for the comparison and the evaluation of different strategies of crowd management and of different structural changes on the geometry of the environment.

Keywords: Multi-Agent Systems, Complex Systems, Crowd, Groups.

1 Introduction

The Multi-Agent Systems approach to the modelling and simulation of complex systems has been applied in very different contexts, ranging from the study of social systems [9], to biological systems (see, e.g., [17]), and it is considered as one of the most successful perspectives of agent-based computing [13], even if it is still relatively young, compared, for instance, to analytical equation-based modelling. Models for the simulation of pedestrian dynamics and crowds have already been successfully applied to several scenarios and case studies, off-the-shelf simulators can be found on the market and they are commonly employed by end-user and consultancy companies[1]. Most of these models employ an agent-based approach, even if the notion of agent does not always relate to the one currently adopted in the autonomous agents and multi-agent systems research area: in fact, several models represent individuals, not aggregate quantities, but they interpret pedestrians as particles subject to attractive or repulsive forces. Despite the substantial amount of results this area is still quite lively: one of the least studied and understood aspects of crowds of pedestrians is represented by the implications of the presence of groups [6]. Even if recent works, especially in the agents technology area [3], represent a promising line of research, current results still need a more thorough validation and an analysis of the feasibility of their application to concrete case studies. The aim of this work is to introduce

[1] see http://www.evacmod.net/?q=node/5 for a significant although not necessarily comprehensive list of simulation platforms.

Y. Demazeau et al. (Eds.): PAAMS 2013, LNAI 7879, pp. 25–36, 2013.

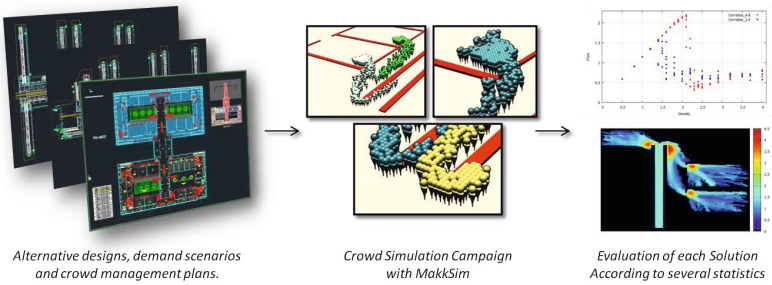


Fig. 1. The application scenario of MAKKSim: evaluation of designs and crowd management solutions by means of statistical data from simulations

MAKKSim, an innovative pedestrian and crowd simulation system considering groups as a central element of the represented scenario. MAKKSim represents an evolutionary step in the direction of an extension of robust currently employed pedestrian models towards more sophisticated behavioural models. In particular, this paper presents a version of the model including an adaptive mechanism for the preservation of group cohesion that represent a significant improvement over the previous versions [2].

The work is organised as follows: first of all, the agent-based model will be described, introducing the elements for the definition of environment, pedestrians and mechanism of movement. Then, the analysis of MAKKSim from an architectural point of view will be described (Sect. 3). The application of the platform to a real-word case study will be discussed with reference to the comparison and the evaluation of different crowd management strategies and alternative geometries of the environment for the scenario (Sec. 4). Conclusions and future developments will end the paper.

2 Agent-Based Computational Model

The agent-based computational model underlying MAKKSim will now be briefly introduced, focusing on the definition of *environment*, *pedestrians* and *movement mechanism*. A more thorough formal definition can be found in [18].

2.1 Environment

The environment is modelled in a discrete way by representing it as a grid of squared cells with 40 cm side (according to the average area occupied by a pedestrian [20]). Cells have a state indicating the fact that they are vacant, occupied by an obstacle or by a pedestrian.

A simulation scenario encompasses both the structure of the environment and all the information required for the realisation of a specific simulation, such as the demands (pedestrians generation profile, origin-destination matrices) and spatial constraints (e.g. crowd management policies). The information related to the scenario of the simulation are represented by means of *spatial markers*, special sets of cells that describe relevant elements in the environment. In particular, three kinds of spatial markers are defined: (i) *start* areas, that indicate the generation point of agents[2] in the scenario, all at once or according to a user defined *frequency*; (ii) *destination* areas, that define the possible targets of the pedestrians in the environment; (iii) *obstacles*, that identify all the non-walkable areas as walls and zones where pedestrians can not enter.

Space annotation allows the definition of virtual grids containing information about agents' positions and movements in the environment. In our model, we adopt the *floor field* approach [4], that is based on the generation of a set of superimposed grids (similar to the grid of the environment) starting from the information derived from spatial markers. Floor field values are spread on the grid as a gradient used to support pedestrians in the navigation of the environment: path fields show the shortest path towards a given destination, density fields highlight instead crowded points of the environment, and so on. Some of the floor fields are *static* (creating at the beginning and not changing during the simulation) or *dynamic* (changing during the simulation). Three floor fields are considered in the model: (i) *path field* assigned to each destination area, that indicates for every cell the distance from the destination, acting as a potential field that drive pedestrians towards it (static); (ii) *obstacles field*, that indicates for every cell the distance from an obstacle or a wall (static);(iii) *density field*, that indicates for each cell the pedestrian density in the surroundings at the current time-step (dynamic).

Chessboard metric with $\sqrt{2}$ variation over corners [11] is used to produce the spreading of the information in the path and obstacle fields; differently, for the density field, when a pedestrian p moves in a cell c, it is modified adding 1 to the cell in which he/she moves, and subtracting 1 from the cell he/she just left. For the neighbour cells, the value v decreases with the inverse of the square of the distance d between the cell and p ($v = \frac{1}{d^2}$).

2.2 Time and Update Mechanism

In our model, time is also discrete, divided into steps of intervals equal to 1/3 seconds. These configurations, along with the adoption of a Moore neighbourhood with radius equal to 1, generates a linear pedestrian speed of 1.2 ms^{-1}, in line with the data from the literature representing observations of crowd in normal conditions [20].

Regarding the update mechanism, three different strategies are usually used in this context [10]: *sequential*, *shuffled sequential* and *parallel* update. The first

[2] Different types of agents with different features can be generated by the same start area.

strategy is based on a sequential update according to a static list of priority, that reflects the generation order of the agents; on the contrary, the parallel update calculates the choice of movement of all the pedestrians at the same time. We currently adopt the shuffled sequential strategy, in which a dynamic list of priority among agents is randomly generated every step and then used to sequentially update agents positions.

2.3 Pedestrians and Movement

As we previously said, pedestrians are defined as utility-based agents:

$$Ped : \langle Id,\ Group,\ State \rangle; \quad State : \langle position,\ oldDir,\ Dest \rangle$$

with their own numerical identifier, their group (if any) and their internal state, that defines the current position of the agent, the previous movement and the final destination, associated to the relative path field.

Agent life-cycle is divided in four steps: *perception, utility calculation, action choice* and *movement*. The *perception* step provides to the agent all the information (i.e., values extracted from floor fields) he/she needs for choosing his/her destination cell. This choice is based on an utility value assigned to every possible movement according to the following function:

$$U(c) = \frac{\kappa_g G(c) + \kappa_{ob} Ob(c) + \kappa_s S(c) + \kappa_c C(c) + \kappa_i I(c) + \kappa_d D(c) + \kappa_{ov} Ov(c)}{d}$$

Function $U(c)$ takes into account all the behavioural elements relevant for pedestrian movement, combining information derived by local floor fields: *goal attraction* (i), *geometric* (ii) and *social* repulsion (iii) are the first factors considered in modelling pedestrian behaviour. In addiction, we introduce two different components to model social relationships between pedestrians: *simple group* (iv), that indicates a family, friends, or any other group characterised by a strong cohesion; *structured group* (v), generally a large one (e.g. group of team supporters), that shows a slight cohesion and a natural fragmentation into subgroups, in which the cohesion gets stronger. Moreover, two factors represent preferences with respect to movement, helping the model to reproduce realistic simulation both

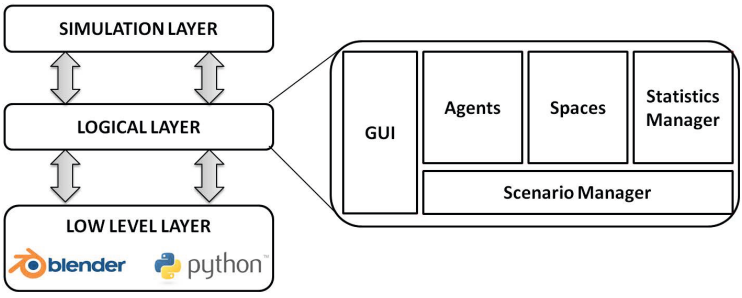


Fig. 2. High-level perspective of MAKKSim architecture

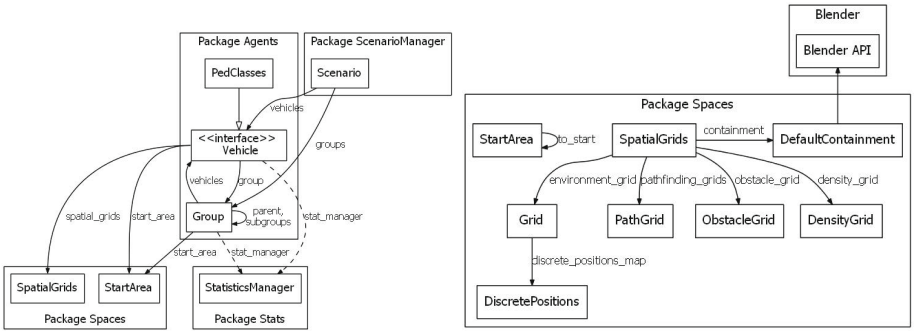


Fig. 3. The class diagram of MAKKSim, from the perspective of *Agents (a)* and *Spaces (b)* packages. Note that in *(b)* links from external packages have been omitted.

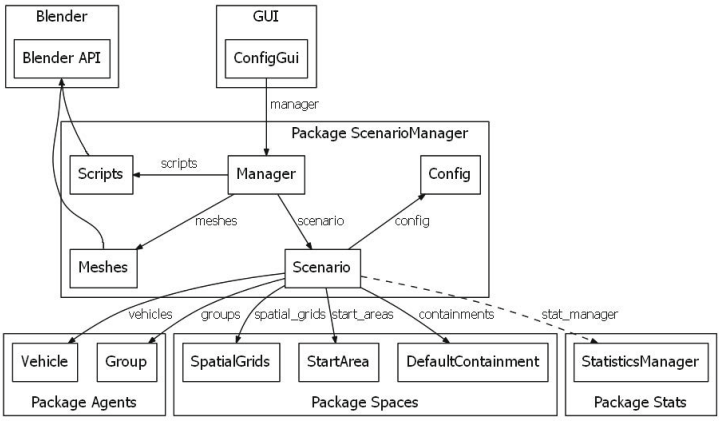
in qualitative and quantitative perspective: (vi) adds a bonus to the utility of the cell next to the agent according to his/her previous direction, while (vii) describes the *overlapping* mechanism, a method used to allow our model the possibility to treat high density situations, allowing two pedestrians occupying temporarily the same cell at the same step. The model includes an adaptive mechanism, modulating the constants that regulate the relative importance of the behavioural components, in particular to preserve the cohesion of simple groups whenever their dispersion grows above a given threshold. More details about this mechanism can be found in [18].

After the utility evaluation for all the cells in the neighbourhood, the choice of action is stochastic, with the probability to move in each cell c as (N is the normalisation factor): $P(c) = N \cdot e^{U(c)}$. The set of possible actions is defined as the list of the eight possible movements in the Moore neighbourhood, plus the action to keep the position (indicated as X): $A = \{NW, N, NE, W, X, E, SW, S, SE\}$.

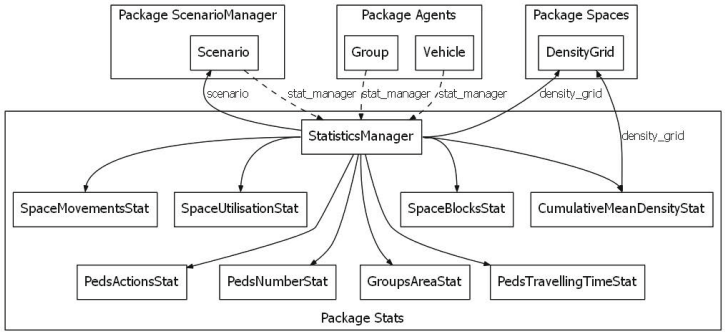
3 System Architecture

Figure 2 shows the high-level architecture of our platform. Users interact with the software through the *simulation* layer, which represents the 3D model of the environment, built using Blender[3], and the information of the simulation scenario, set by MAKKSim graphic interface. The *logical* layer contains the Python code of the simulator, structured in 5 packages: the *GUI* package (i), the *Agents* (ii) and *Spaces* packages (iii), that constitute the implementation of the computational model previously described, the *Scenario Manager* (iv), which connects the packages to run the simulation and the *Statistics Manager* (v), for statistics purposes. The last layer is the *low level* layer, that contains both the

[3] Blender is a free and open-source 3D computer graphics software, `http://www.blender.org`



(a)



(b)

Fig. 4. The class diagram of MAKKSim, from the perspective of *ScenarioManager (a)* and *Stats (b)* packages

Python virtual machine, which runs the code, and the Blender 3D engine, that supports an easy-to-read visualisation of the results of the simulations.

A more detailed explanation of the main software packages will be now presented, by means of the UML-like class diagram showed in Figure 3 and 4.

***Agents* Package.** The package *Agents* represents the portion of the software that implements the pedestrians. In order to allow the possibility to define heterogeneous types of pedestrians, the basic schema of an agent has been implemented starting from the interface *Vehicle*, in which there are four abstract methods that exactly represent the agent life-cycle: *perceive*, *evaluate*, *choose* and *update*. With this methodology, to develop a new kind of pedestrian, it is just necessary to create a new class that implements the *vehicle* interface - or extends one of the already-defined pedestrian class - and override these methods. For the representation of the element *group* class with the same name has been

defined. In case of simple group, it will contain only a set of agents, otherwise it will be composed of the list of sub-groups.

***Spaces* Package.** To link the definition of pedestrians with the environment of the simulation, every pedestrian instance has the references to the main elements of the package *Spaces*: *SpatialGrids*, which represents the container of the spatial grids (including the floor fields); the relative *Start* marker of its generation. Considering the management of the environment and spatial markers, floor fields are defined respectively by the classes *PathGrid*, *ObstacleGrid* and *DensityGrid*, while the grid of the discrete environment is implemented by *Grid*, and *DiscretePositions* aggregates the positions of all pedestrians. The marker *Start* has been implemented as a class due to its complexity: it defines the generation frequency of pedestrians and groups, the type of pedestrians, the stochastic assignment of goal to agents and, finally, the reference to another *start area* to indicate where the agents will be regenerated after the goal reaching (the latter is useful to realise agents with multiple tasks or subjected to vertical movements). The interaction between MAKKSim and the Blender engine is managed by the class *DefaultContainment* that acts as a "mediator" between the continuous environment of Blender and the discrete one of MAKKSim, for the creation of the spatial grids.

Scenario Manager. In order to perform simulations, the main elements of the two previous packages are initialized and connected by the *ScenarioManager*, specifically by the object *Scenario* (Figure 4(a)). The latter is controlled by *Manager*, which is the root class of the software and runs the simulations by initialising and updating *Scenario*. Its initialisation requires the object *Configuration*, created by the GUI class with the parameters of the simulation, defined by the user. At each step the class *Scenario* is therefore updated, both with the introduction of further agents in the simulation (according to the configured frequencies) and the update of those that are already in the scenario. The elements *Meshes* and *Scripts* implement integration points with Blender: the first one is used to draw simple 3D model of the pedestrians, while the other one integrates the Python code of MAKKSim in Blender.

Statistics Manager. Class *Scenario* initialises also the core of the package *Stats*, named *StatisticsManager*, that supports the definition of statistics. We chose to implement this one as a Singleton, in reference to the well-known software architecture design patterns [8], due to the fact that it is an object which must be reachable from all agents of the simulation (and the Singleton reference can be obtained, instead of saved in each of them[4]) and, moreover, it is strictly required that only one instance of it is created.

Therefore the *StatisticsManager* works as the mediator between the agents and the statistics management objects, for the realisation of the statistical data. Classes into *StatisticsManager* control have been implemented to perform, for every step, different kind of analysis:

[4] It is the reason why in the Figure 3 and 4 the links to *StatisticsManager* are dashed.

• *SpaceUtilizationStat* saves the absolute values of cell utilisation during all the simulation, using a special discrete grid in which 1 is added in every occupied cell. It allows to analyse the average trajectories of pedestrians;
• *SpaceMovementsStat* and *SpaceBlocksStat*, similar to the previous one, focusing on the pedestrian real movements and on block situations, respectively, by adding 1 in the corresponding cells;
• *CumulativeMeanDensityStat* calculates the grid of cumulative mean density (CMD) of pedestrians[5]. This grid is obtained as follows: agents store the value of the *density field* in its position in the grid and at the end of the simulation the mean value is then calculated. Note that, with this method, CMD values of empty cells gets no variation, therefore the grid does not vary on time;
• *PedsNumberStat* saves the number of agents in the simulation;
• *PedsActionsStat* stores every action performed by the agents during all the simulation, to analyse their trajectories;
• *PedsTravellingTimeStat* stores the number of steps required by agents to reach their goal;
• *GroupsAreaStat* stores the values of pedestrian groups dispersion during the simulation. Group dispersion is a fuzzy concept and different methods can be defined to calculate it [1]. Our method identifies the dispersion of a group as the area of the convex hull generated using the position of its members.

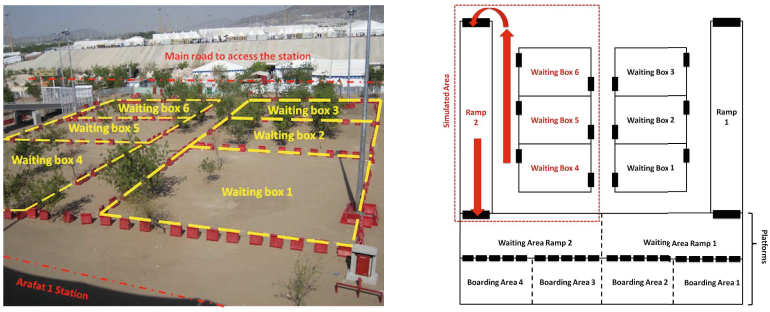
Results of the analyses are saved both as comma separated values (CSV) files (for further elaboration into a spreadsheet or plotting through tools like GNU-Plot) and as portable network graphics (PNG) images, when possible and relevant, for an immediate visual feedback.

4 Application Scenarios and First Results

Before describing and analysing the results, however, we will introduce some considerations about the computational costs of the simulations, as well as about model calibration and validation. Simulations have been performed using a workstation with an Intel(R) Xeon(R) E5630 CPU (12 MB SmartCache, 2.53 GHz clock), 6 GB Ram and Windows 7 Professional 64 bit as OS. Tests have shown up the significant computational complexity of the agent behavioural model: simulations performed in the Arafat Station scenario, that will be described later on, include a population of 750 agents, divided into three equal structured groups, and they required an average time of 14 seconds per simulation step[6].

[5] CMD is defined as the average local pedestrian density perceived from each person in each of his/her positions (see [5] for a detailed explanation).

[6] It is clear that computational time is a significant aspect with respect to the use of simulation platforms, but, during the design of the tool, it was not our aim to deal with real time simulations, that however cannot be achieved even with commercial systems but only with prototype research tools [19] and in relatively simple situations, from the point of view of pedestrian behaviour complexity. Despite that, works on optimization and parallelization of different tasks, like the statistics calculation, can be developed in the future.



(a) (b)

Fig. 5. Picture of the waiting boxes taken from the Arafat station (a) and the logical representation of the simulation scenario (b)

Regarding model calibration and validation, a set of preliminary tests on benchmark scenarios[7] have been performed, with the aim of understanding the platform capability to reproduce the aggregated macroscopic effects of the pedestrian dynamics [15]. By comparing the simulated data with the empirical evidences, we have obtained ranges of values for all the parameters in $U(c)$ function (see Sec. 2.3), allowing to perform realistic pedestrian behaviours and simulations; more details on these experimental scenarios can be found in [18]. We want to emphasise that it is this aim of producing results that are comparable to real data in a wide range of situations that calls for a relatively simple model: much richer behavioural models are present in the literature (see, e.e., [16]), but they are mostly aimed at producing visually realistic animations of environments.

The first real world application of MAKKSim was in the context of CRYSTALS[8] project: several tests have been performed on the Arafat I station on the Mashaer Metro line in Saudi Arabia to compare and analyse various solutions of crowd management, in a situation that involves large amount of pedestrians. In fact, the station is used during the Pilgrimage towards Mecca, moving a large number of people from different holy sites in a strict period of time. This scenario focuses on the entrance to the station (Figure 5). In 2010 the crowd management procedure applied in the scenario was the follows: pilgrims were divided into groups with 250 members, which were firstly directed in special zones called *waiting boxes*. The access to the ramp from waiting boxes was therefore allowed one group at a time, in order to maintain a comfortable pedestrian flow and an acceptable density on the ramp and inside the station. With the use of MAKKSim we have explored particular worst cases, in order to evaluate potential changes in the crowd management procedure, or to the environmental structure of the station.

[7] These tests had the goal to evaluate, both in quantitative and qualitative way, the model effectiveness to reproduce well-known crowd behaviours, by a set of simulations performed in simple environments, like corridors or junctions, due to the presence of sufficient empirical data available [12].

[8] `http://www.csai.disco.unimib.it/CSAI/CRYSTALS/`

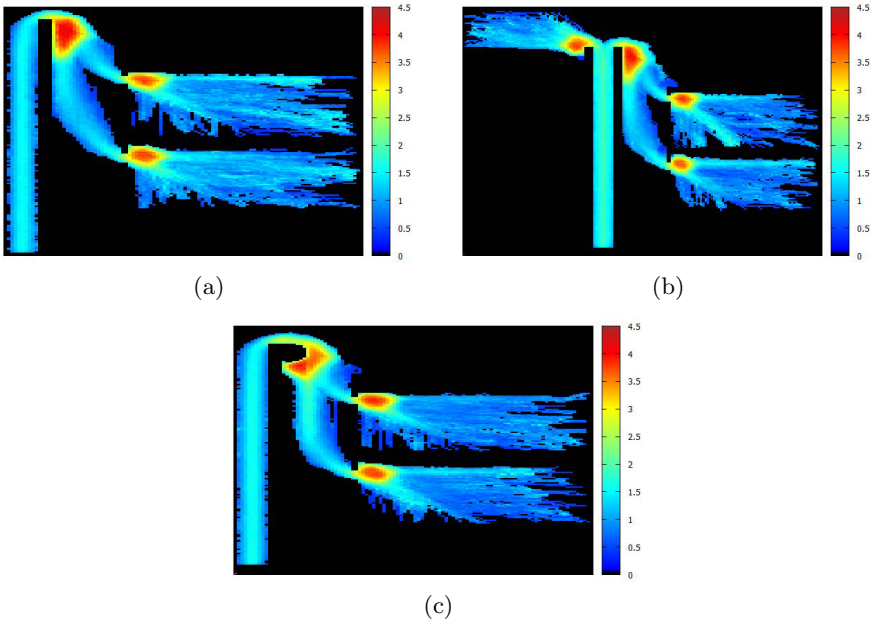


Fig. 6. CMD maps of the simulations on Arafat I station, in the three different scenarios

With the aim to restrict simulation time, we simplify the scenario by focusing on the inflow of 3 waiting boxes to the station and assuming that these ones already host one group. Every waiting box overlaps with a start area, which generates a block of 250 pilgrims subdivided in several groups, with the goal to reach the station passing through the ramp (identified as *simulated area* in Fig. 5 (b)). Starting from this configuration, three scenarios have been defined, simulated and analysed: (i) the case in which *two waiting boxes* are activated simultaneously, to verify if the parallelisation of the use of the ramp by two groups is plausible, taking into account both the ingress time and the comfort of the pilgrims; (ii) a similar situation in which an additional *external group of 250 pilgrims*, generated on the other side of the ramp, joins the groups from the waiting boxes to simulate the case in which the crowd management procedures are not respected; (iii) the case of simultaneous flow from two waiting boxes, with a *change in the geometry* of the station, introducing a large obstacle at the entrance of the ramp.

Some of the statistical results of the simulations are shown in Figure 6, representing maps of the CMD of the three scenarios. In Figure 6(a), it is possible to note that the simultaneous activation of two waiting boxes lead to a congestion near the entrance of the ramp, with a very high perceived pedestrian density (the maximum value of 4.5 m^{-2} has been reached). The total time of the simulation is improved of 10% respect with the sequential used of waiting boxes; despite that, taking into account the comfort variable, the density reached is considered not acceptable for the pedestrian walkways standard [7], so this solution should

not be applied in this context. About the total time of ingress, the result of the second scenario is similar to the previous one. The problem of high densities still characterises this situation, that represents the worst case of our simulations: in Fig. 6(b) the external flow increases the density inside the ramp, that reaches a value near to 2.5 m^{-2}, also considered not acceptable. In the third scenario we try to improve the congestion situation near the entrance of the ramp, generated by the simultaneous activation of the flow from two waiting boxes: as suggested in [14] we tested the introduction of an obstacle near the most crowded point to see if this could cause a smoothening of the flow. Fig. 6 (c) shows that the round obstacle modelled leads to lower values of CMD near the entrance of the ramp. Furthermore, the average time of the simulation is decreased by 3%, because the obstacle has helped the pedestrians to better distribute themselves on the space, improving the flow.

5 Conclusions and Future Works

The paper has introduced MAKKSim, an innovative agent-based pedestrian and crowd simulation system including groups as a central element influencing system dynamics. The paper has briefly introduced the model on which the system is based, then it has shown the overall system architecture and finally a real world application based on MAKKSim has been discussed. Future works are aimed at further validating the model in additional experimental and real world scenarios, especially the group cohesion component of pedestrian's behavioural model; in addition, we intend to extend the model and the system to improve its practical applicability in more complex scenarios, modelling additional phenomenologies and practical environmental infrastructures. Finally, additional work must be done to improve the efficiency of the simulation, in particular through the potential parallelisation of some tasks.

References

1. Bandini, S., Rubagotti, F., Vizzari, G., Shimura, K.: An agent model of pedestrian and group dynamics: Experiments on group cohesion. In: Pirrone, R., Sorbello, F. (eds.) AI*IA 2011. LNCS, vol. 6934, pp. 104–116. Springer, Heidelberg (2011)
2. Bonomi, A., Manenti, L., Manzoni, S., Vizzari, G.: Makksim: Dealing with pedestrian groups in mas-based crowd simulation. In: WOA. CEUR Workshop Proceedings, vol. 741, pp. 166–170. CEUR-WS.org (2011)
3. Bosse, T., Hoogendoorn, M., Klein, M.C.A., Treur, J., van der Wal, C.N., van Wissen, A.: Modelling collective decision making in groups and crowds: Integrating social contagion and interacting emotions, beliefs and intentions. Autonomous Agents and Multi-Agent Systems 27(1), 52–84 (2013)
4. Burstedde, C., Klauck, K., Schadschneider, A., Zittartz, J.: Simulation of pedestrian dynamics using a two-dimensional cellular automaton. Physica A: Statistical Mechanics and its Applications 295(3-4), 507–525 (2001)
5. Castle, C., Waterson, N., Pellissier, E., Bail, S.: A comparison of grid-based and continuous space pedestrian modelling software: Analysis of two uk train stations. In: Pedestrian and Evacuation Dynamics, pp. 433–446. Springer US (2011)

6. Challenger, R., Clegg, C.W., Robinson, M.A.: Understanding crowd behaviours: Supporting evidence. Tech. rep., University of Leeds (2009)
7. Fruin, J.J.: Pedestrian Planning and Design. Metropolitan Association of Urban Designers and Environmental Planners, New York (1971)
8. Gamma, E., Vlissides, J., Helm, R., Johnson, R.: Design patterns: Elements of reusable object-oriented software. Addison - Wesley (1995)
9. Gilbert, N., Troitzsch, K.G.: Simulation for the Social Scientist, 2nd edn. Open University Press (2005)
10. Klüpfel, H.: A Cellular Automaton Model for Crowd Movement and Egress Simulation. Ph.D. thesis, University Duisburg-Essen (2003)
11. Kretz, T., Bönisch, C., Vortisch, P.: Comparison of various methods for the calculation of the distance potential field. In: Pedestrian and Evacuation Dynamics 2008, pp. 335–346. Springer, Heidelberg (2010)
12. Kretz, T., Grünebohm, A., Kaufman, M., Mazur, F., Schreckenberg, M.: Experimental study of pedestrian counterflow in a corridor. Journal of Statistical Mechanics: Theory and Experiment 2006(10), P10001 (2006)
13. Luck, M., McBurney, P., Sheory, O., Willmott, S. (eds.): Agent Technology: Computing as Interaction. University of Southampton (2005)
14. Nishinari, K., Suma, Y., Yanagisawa, D., Tomoeda, A., Kimura, A., Nishi, R.: Toward Smooth Movement of Crowds. In: Pedestrian and Evacuation Dynamics 2008, pp. 293–308. Springer, Heidelberg (2010)
15. Schadschneider, A., Klingsch, W., Kluepfel, H., Kretz, T., Rogsch, C., Seyfried, A.: Evacuation Dynamics: Empirical Results, Modeling and Applications. ArXiv e-prints (2008)
16. Shao, W., Terzopoulos, D.: Autonomous pedestrians. Graphical Models 69(5-6), 246–274 (2007)
17. Textor, J., Hansen, B.: Hybrid simulation algorithms for an agent-based model of the immune response. Cybernetics and Systems 40(5), 390–417 (2009)
18. Vizzari, G., Manenti, L., Crociani, L.: Adaptive pedestrian behaviour for the preservation of group cohesion. Complex Adaptive Systems Modeling (in press 2013)
19. Wagoum, A.U.K., Chraibi, M., Mehlich, J., Seyfried, A., Schadschneider, A.: Efficient and validated simulation of crowds for an evacuation assistant. Journal of Visualization and Computer Animation 23(1), 3–15 (2012)
20. Weidmann, U.: Transporttechnik der fussgãnger: transporttechnische eigenschaften des fussgãngerverkehrs (literaturauswertung) (1993), http://infoscience.epfl.ch/record/41377