

Proceeding of the 11th [World Conference on Transport Research \(WCTR\)](#), University of California, Berkeley, 2007

A NAVIGATION ALGORITHM FOR PEDESTRIAN SIMULATION IN DYNAMIC ENVIRONMENTS

Kardi Teknomo¹ and Alexandra Millionig²

¹ Ir., M.Eng, Ph.D., Senior Researcher, Arsenal Research, Austria
(teknomo@gmail.com)

² DI., Researcher, Vienna University of Technology, Austria
Alexandra.Millionig@arsenal.ac.at

Address:

Arsenal Research, Giefinggasse 2, Vienna 1210, Austria

A NAVIGATION ALGORITHM FOR PEDESTRIAN SIMULATION IN DYNAMIC ENVIRONMENTS

Kardi Teknomo¹ and Alexandra Millonig²

¹Arsenal Research, Austria

²Vienna University of Technology, Austria

ABSTRACT

In this paper we introduce a novel online path planning method that mimics the human sense of navigation for finding a path between the current location (the ‘origin’) and the destination point. Most pedestrian simulation systems pre-compute a minimum cost (e.g. shortest length) path. Such an approach implies that the simulated agents have perfect knowledge of the infrastructure. Moreover, pre-computing the path can be numerically infeasible in situations where some of the obstacles are of dynamic nature, e.g. doors that might be closed or open, escalators which might operate or be under repair or obscure regions of parks which pedestrians choose not to frequent at night time. For such dynamic environments, the shortest path needs to be pre-computed for every - out of many - possible combination. The algorithm proposed in this paper assumes that the modelled agent may or may not have any prior knowledge of the infrastructure but is able to locate the direction of the destination at all times. The proposed navigation algorithm is an iterative procedure to maximize the values of smoothed distance and directional function from the destination to the origin. Such an algorithm is guaranteed to find a path from the origin to the destination if such a path exists at all. The proposed method has been implemented in a pedestrian multi agent simulation and has been tested on scenarios with different complexities.

Keywords: wayfinding, multi agents, mesoscopic, relaxation, sink propagation value

INTRODUCTION

In this paper we consider the problem of finding a path from an entry point to a target point in a dynamic infrastructure in the context of pedestrian simulation. Pedestrian simulation is a representation of pedestrian movement using a set of mathematical models that can be used to evaluate various designs of pedestrian facilities during the planning stage. The term 'infrastructure' is used for public infrastructures such as train stations, airports, shopping centres and also for parks and areas of a town (such as e.g. town districts). 'Dynamic' refers to different configurations of the considered infrastructure resulting from the presence or absence of mobile obstructions (e.g. open or closed doors, closed walkways or parks with gate opening hours in the case of larger regions considered). We consider a dynamic environment consisting of several obstructions, in which some obstructions can be dynamically changed (e.g. doors can be open and close, broken escalator need to be repaired for some time, etc.).

This problem arises in the context of agent based models for pedestrian motion. Recently, most of the prominent pedestrian simulation models are mainly based on either Cellular Automata [Kretz and Schreckenberg (2006), Schadschneider (2001), Blue and Adler (2000)] or Force model [Helbing and Molnár (1995), Teknomo and Gerilla, (2005)]. These simulation models, however, do not consider pedestrian navigation. Such simulation models are working very well only to simulate pedestrians in single or double rooms with simple obstructions. In the absence of obstructions such as walls or a room with a table in the middle, etc. pedestrian agents can move forward and avoid simple obstructions; still they cannot find a way to the destination if the destination is hidden behind an obstruction. Only based on such models alone, pedestrian agents will be able to move from origin cells to destination cells if and only if the line connecting origin and destination (OD line) does not pass any obstruction. To simulate a more realistic and complex infrastructure, however, pedestrian agents should be able to navigate from one place to another in the presence of obstructions. Hence, a navigation algorithm is needed.

Path planning in a static environment can be computed by many methods such as region decomposition with the visibility graph, navigation grid, shortest distance map or dynamic programming, roadmap or skeleton, landmark based navigation, depth first tree and potential field [e.g. Lee (1983), Latombe (1995), Hershberger and Suri (1999)]. Usually, wayfinding tasks are solved based on the assumption that the agents always use the shortest path to reach their aim. Hoogendoorn and Bovy (2004), for example, include navigation in their level of strategic planning where the path is always assumed to be chosen optimal according to a specific criterion function. This optimization implies that all information is available from the outset. Moreover, they only deal with the static case where the geometry of the considered infrastructure is fixed and known to the agents. In this standard setting, the agents determine their path typically by minimizing some cost function such as the length of the path or the walking time. However, there are many settings in which it is unrealistic to assume that the agent immediately finds the shortest path. Such a situation arises, for instance, when using a hiking map providing only imperfect information about the possible

paths, or when searching for the exit in the main hall of a train station which is solely marked with a sign attached high on a wall above the exit.

Our approach differs from this static, cost optimizing navigation in two aspects: First we consider a dynamic environment where many obstructions potentially vary. A quick re-computation of the cost of the optimal route under all different possible configurations becomes infeasible since the number of combinations equals 2^N where N denotes the number of dynamic obstructions. Thus, for five dynamic obstacles 32 different configurations have to be calculated. Secondly, we do not rely on the assumption that the pedestrian agents possess a map of the infrastructure available nor have prior knowledge and experience to use the infrastructure well at the time of entry. In fact, this paper tries to supply a novel algorithm for wayfinding in which the model includes a degree of information parameter to cope with variations of prior knowledge of the infrastructures. The infrastructure can be assumed as either initially unknown, partially known to some degree or fully recognized by the subject trying to find her way.

The remaining article is outlined as follow. The next section provides a brief discussion of the mesoscopic pedestrian multi agent simulation model and how the environment is modelled spatially. Then, in the Dynamic Navigation section we explain our navigation algorithm for both static and dynamic environment. First we describe the navigation matrix for static environments, and then we generalize the same navigation matrix for a dynamic environment. Before the conclusions, we demonstrate the navigations for dynamic environment through example of scenarios to investigate the effect of degree of information toward the infrastructure.

PEDESTRIAN MULTI AGENT SYSTEM

The basic pedestrian simulation model used in this article is a state based spatial model of cellular automata (CA). Time and space are discrete and the movements of pedestrian agents are described by a set of rules. Wolfram (2002) stated that many CA models are equivalent to partial differential equations. However, CA represents a larger set of dynamical systems as the rules are not necessarily in algebraic form.

The environment is modelled as a regular lattice and the space is spatially decomposed into cells. For each pedestrian agent, each cell represents a state of being at a particular location and the next state is restricted only to the nine neighbouring cells including the current state as illustrated in Figure 1. These nine neighbouring cells are often called *Moore neighbourhood* (Wolfram (2002)). The layout plan of the virtual environment is divided into grid cells of a certain diameter (e.g. about 0.5 meter up to 2 meters) and represented in the model as a binary matrix. Zero value in the matrix represents a wall or other obstruction and the value of one indicates a free space for the agents to occupy and move around. The binary values also emphasize the permissible state since pedestrian agents are not allowed to consider an obstruction cell as a possible next state. Without losing generality, the measurement to move around uses the chessboard distance instead of the Euclidean distance to ease the computation since diagonal positions are not considered as different distances.

The cell diameter is measured as an average diameter of the inner circle and the outer circle of each grid cell.

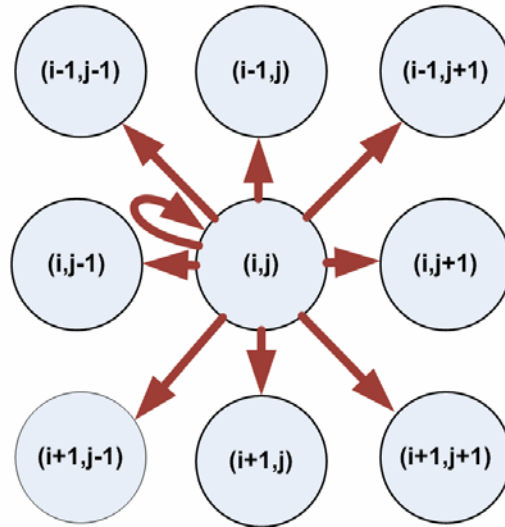


Figure 1. Possible next state transitions

Different from the existing pedestrian microscopic CA models [e.g. Blue and Adler (2000), Schadschneider (2001), Kretz and Schreckenberg (2006)], our model does not impose the restriction that a cell can be occupied only by a single pedestrian agent. In fact, the model generalizes microscopic CA in a way that a cell can be occupied by any number of agents up to the maximum density of a cell. A special case occurs when we use a diameter of 0.5 meters and the probability to enter the next cell is one if there is no pedestrian in that cell, and the probability equals zero if the cell is occupied. By employing a restriction of one cell for one agent, we will regain the microscopic CA model. In this sense, our mesoscopic model forms a kind of generalization of pedestrian simulations using the CA model.

Table 1. Probability to enter next cell as function of cell density

Number of pedestrians in a cell	Probability to enter this cell for another pedestrian
0	1.0
1	0.8
2	0.6
3	0.4
4	0.2
5	0.0

The interactions among pedestrian agents are described by two functions of density which are imposed for each lattice cell. The first function refers to the probability to enter the next cell, while the second function refers to the speed density relationship. These two functions can be displayed in tables. Table 1 shows an example of the probability function to enter the next cells in the neighbourhood for a cell diameter of 1 meter. The relationship is linear up to a maximum density of 5 pedestrians per cell. An increasing amount of pedestrians situated in a certain cell will decrease the probability of an additional pedestrian to enter this particular cell. Neufert (2000) provides a rough guideline concerning pedestrians moving at normal speed, which

specifies the space they occupy with an area of approximately 75 cm by 62.5 cm for foot placement, not including the space between pedestrians. This equates the area module of 0.47 square meters or a density of about 4 to 5 pedestrians per square meter.

The agent's timing to go out of a cell is influenced by the cell diameter and the agent's current speed while the current agent's speed is a factor of density. An example of the linear speed density relationship is outlined in Table 2 for the same cell diameter of one meter. The maximum density is 5 pedestrians per cell and the maximum speed is 1.4 meter/seconds.

Table 2. Linear Speed-Density Relationship

Density (number of pedestrian per cell area)	Speed in m/s
0	1.40
1	1.12
2	0.84
3	0.56
4	0.28
5	0.00

If u denotes the speed in meter/second and k is the density (represented by a discrete number of pedestrians per cell area), the theoretical relationship between speed and density can be generalized by setting it as an approximation function of a cumulative Beta distribution.

$$u = 1 - \frac{B_k(\alpha, \beta)}{B(\alpha, \beta)} \quad (1)$$

The nominator $B_x(\alpha, \beta)$ is the Incomplete Beta Function formulated as

$$B_x(\alpha, \beta) = \int_0^x g^{\alpha-1} (1-g)^{\beta-1} dg \quad (2)$$

The denominator $B(\alpha, \beta)$ is the Beta function formulated similar to the Incomplete Beta function with $x = 1$, which is

$$B(\alpha, \beta) = \int_0^1 g^{\alpha-1} (1-g)^{\beta-1} dg \quad (3).$$

The range of two parameters is $\alpha > 0$ and $\beta > 0$. The linear function is a special case when $\alpha = \beta = 1$.

Each pedestrian is modelled as an agent that keeps his or her own properties, such as the time to enter and leave a cell, the agent's speed and the tracked trajectory. The movement vector of each agent is selected based on the maximum argument of a normalized neighbourhood function. A neighbourhood function is a Moore neighbourhood subset of a matrix with the current position of a pedestrian agent located in the centre. Taking the neighbourhood function of three matrices – the binary layout matrix, the matrix probability to enter a cell and the navigation matrix – and multiplying them as a Hadamard product, followed by normalizing the product to the range of [0,1] produces the normalized neighbourhood function. The navigation

matrix will be explained more detailed in the next section. The normalization is necessary to avoid a bias of higher values in the navigation matrix.

DYNAMIC NAVIGATION

The navigation principle aims at finding the next higher state value. Here, the navigation is similar to optimization methods such as hill climbing. Instead of finding the best route to the top of the hill function, the navigation of the agents is simply done by using the maximum neighbourhood values. The problem now is to find a hill function which has continuously increasing values from any neighbourhood states to a specified destination location.

As mentioned in the last section, one of the main factors for pedestrian movements is a navigation matrix. The navigation matrix is a normalized displacement vector function which guides the movements of a pedestrian agent towards the destination. Assuming the origin cell being a source basin and destination cell being a sink basin in a dynamical system, and assuming that all neighbourhood cells are connected from the source basin to the sink basin, we can model the values of the navigation matrix as a function of distance and the direction of any cell towards the destination cell (i.e. sink basin). All agents originate from the source basins and they will be ‘repelled’ from the source due to the attraction of the sink. Given such a continuously increasing function, all agents will eventually be guaranteed to reach the sink if such a path exists.

Since the values are normalized, the real value of the navigation matrix is not important. It is the comparison value between one cell and the other within a neighbourhood that is essential to guide a pedestrian’s movement. A lower distance between a cell and the destination, and a more direct way to reach the destination, will lead to a higher value in the navigation matrix. Destination cells will have the highest values while all other free cells are smoothed down towards zero. The lowest value of a navigation matrix is associated with the origin cell (i.e. source basin).

At first glance, more advance readers may find similarity between our concept of a navigation matrix and the potential field theory. The similarity happens because both potential field and our navigation matrix uses source basin and obstructions produce repelling fields while the sink basin produce attractive fields. The detailed concept, however, is different. Potential field uses vector representation that any point is space that has magnitude and direction. The problem with potential field comes in adjusting the combination strength of several fields. Sometime the field that pushing away from obstructions can prevent the solution of any path. Local minima make the agent stuck. Adding noise and outward source field is not a guarantee for non-local minima.

In contrast, our concept of navigation matrix is a cellular representation of what we called sink propagation value. Sink propagation value (SPV) is a value on vertex on a network graph that monotonically increasing (or decreasing) from sink vertex according to the minimum distance function from that vertex to the sink. Therefore, the value of navigation matrix is represented by scalar value that only has normalized magnitude at certain point. The direction is implicit, as it is given in the neighbourhood rather than at a certain point in space. The navigation matrix

computation has already contained full navigation information on the static environment such as obstructions and wall. No additional combination of SPV is needed for static environment. The magnitude of SPV is normalized such that combination of SPV is only needed for dynamic environment.

Unlike potential field that require each agent also to have its own potential field, in our model, agent is independent of the SPV thus can be model independently. Local minimum, however, may exist in the navigation matrix only if the sink value is too small or the location cell has no connection path to the sink. The solution is simply to increase the strength of attraction value if the layout is somewhat complicated.

There are many ways to compute a navigation matrix with the above-mentioned properties such as using Q learning or distance transform. We presented here a fast computational method to obtain a navigation matrix through the *smoothing relaxation* method (Winston, 1993).

The smoothing relaxation method arbitrates between smoothness expectations and the actual data based on local constraints. It is a smoothing or numerical interpolation method to fill the gap between measurements based on the confidence level of the data. Given a measurement matrix and a corresponding confidence level matrix ($0 \leq c_i \leq 1$), the measurement data are interpolated based on the values of the four neighbours.

$$a^{k+1}(i, j) = a^k(i, j) \cdot c(i, j) + (1 - c(i, j)) \cdot \frac{a^k(i-1, j) + a^k(i+1, j) + a^k(i, j-1) + a^k(i, j+1)}{4} \quad (4)$$

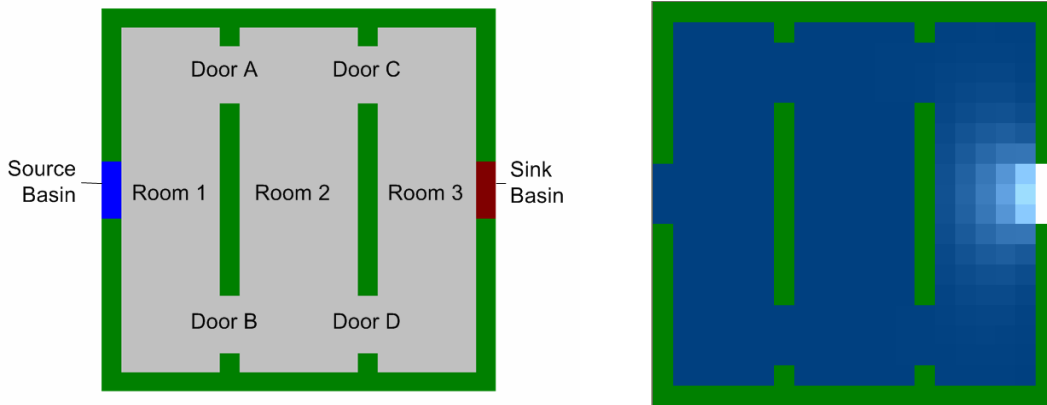


Figure 2. Layout plan of three rooms with open doors (left) and its navigation matrix (right)

The navigation matrix \mathbf{Q} is obtained from the binary matrix that represents the static environment. The confidence level matrix is a binary matrix where sinks and obstruction cells have one confidence levels while a free space has zero confidence level. A zero confidence level indicates no measurements in the free space cells, while a 100% confidence level is assigned to a sink and obstruction to impose the importance of those values. High positive sink values are assigned to ensure the navigation ability of the agents. Low sink values do not provide enough “strength” to allow pedestrians to overcome the obstructions which causes the pedestrian agents to move around some regions without the ability to reach a destination. In general,

setting a high value beyond a certain critical value results in unchanging movement patterns regardless of the values. Figure 2 on the left shows an example of a layout with three rooms connected by four open doors. The right figure shows the navigation matrix of the layout. Lighter colour represents higher navigation value. The navigation value is gradually increasing from the source basin towards the sink in monotonic manner. The highest navigation values occur on the sink basin.

Implicitly, the basic assumption on the navigation matrix described above is perfect information among pedestrian agents to know all the routes. In other words, pedestrian agents are assumed to have apriori knowledge on the infrastructure. Such assumption is only valid for a static environment when the layout plan does not change over time. As emphasized on the introduction, for dynamic environment with many doors opened and closed during the simulation, pre-computation of such combination of navigation matrices is computationally infeasible. To implement such users' interactivity, we need to have another navigation matrix.

Now we would like to introduce a method to navigate in dynamic environments. Instead of using another model for navigation matrix computation, we generalized the navigation algorithm for dynamic environments into the following formulation:

$$\mathbf{Q} = (1 - \gamma) \cdot \mathbf{Q}^{initial} + \gamma \cdot \mathbf{Q}^{current} \quad (5)$$

The computation of a navigation matrix remains the same as described in equation (4) for two different setting in the layout plans. The initial navigation matrix is pre-computed for static environments with all doors open while the current navigation matrix is computed during the simulation according to the current situation of the configurations of the doors. A smoothing parameter $0 \leq \gamma \leq 1$ is added to highlight the degree of information. A higher the value of γ represents a higher degree of information based on the agent's knowledge and previous experience concerning the infrastructure (e.g. is the door in the next room already open or not).

The navigation matrix for dynamic environments is actually a generalization of the navigation in static environments, because – without the existence of any door – the initial navigation matrix is equal to the current navigation matrix and equation (5) produces the same navigation matrix for whatever value of γ .

SIMULATION RESULTS

In this section we illustrate the navigations for dynamic environments through simple scenarios of a layout plan with three rooms, where each room is connected to the next room by a couple of doors as shown in Figure 3. The network graph corresponding to the layout plan is also shown. The behaviour of pedestrian agents near open and closed doors were investigated and compared to a more realistic behaviour that mimics the human sense of navigation.

The first scenario is the base scenario running for static environments where all doors are open. Figure 4 shows three scenes where pedestrian agents select the best route automatically according to the shortest distance (and therefore travel time) to the destination from the source basin (Figure 4(a)), move through door $A \rightarrow C$ and $B \rightarrow D$

(Figure 4(b)) and finally reach the sink basin (Figure 4(c)). There is no flow from door A to door D or from door B to door C. The figures show spatial density: darker cells indicate higher density and lighter cells represent low density.

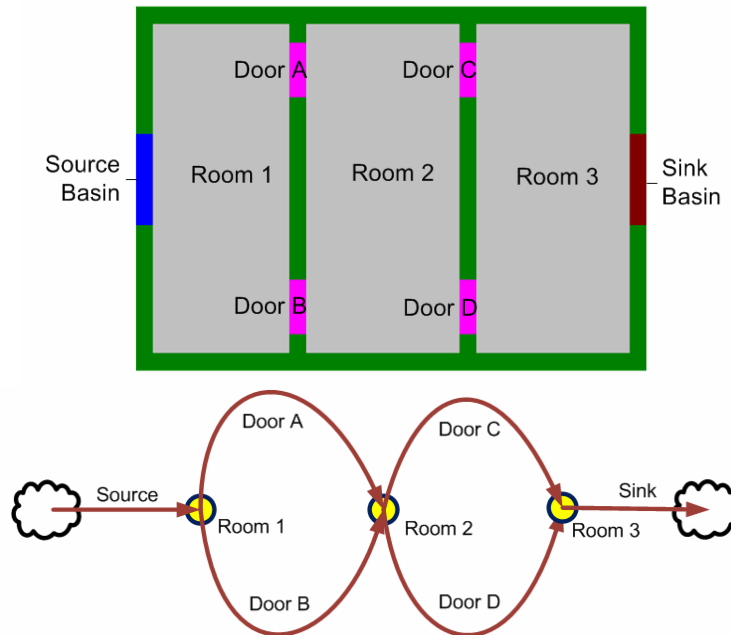


Figure 3. Layout plan of 3 rooms with 4 doors (top) and the network graph associated with the layout (bottom)

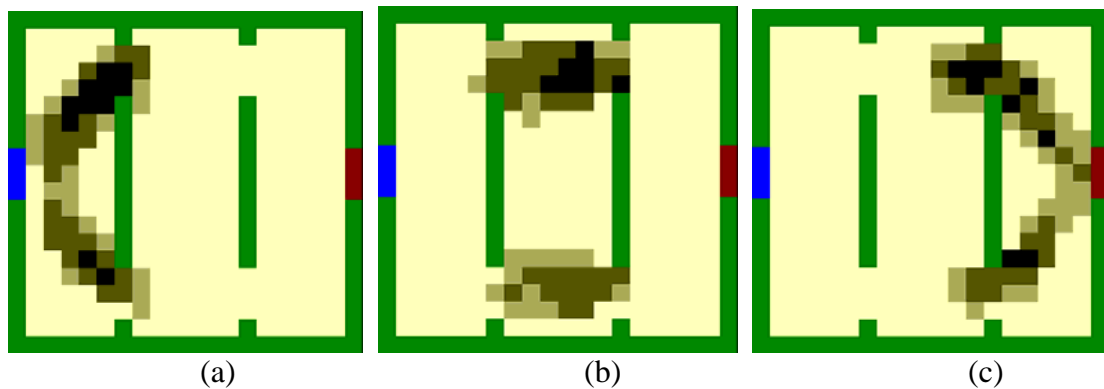


Figure 4. Running simulation in static environment

All scenarios below are designed to illustrate interactivity in dynamic environments. All doors were set to be closed at the beginning of the simulation to demonstrate the behaviour of pedestrian agents for different values of degree of information.

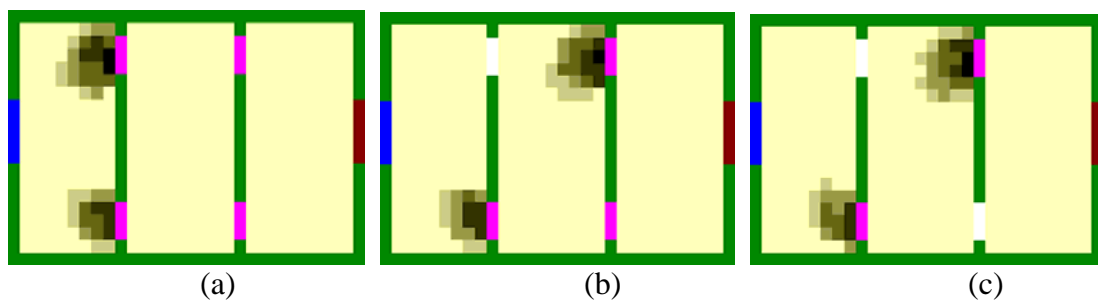


Figure 5. Running simulation in a dynamic environment at no information ($\gamma = 0$)

The second scenario is running in a dynamic environment with no information regarding the infrastructure (i.e. $\gamma = 0$). Figure 5(a) shows that the agents will queue in front of the nearest door in the first room because all doors are closed. The queuing arc similar to real pedestrian behaviour was observed. Then door A that connects the first and second room is opened and the agents who first queued in front of door A are now queuing in front of door C as shown in figure 5(b). Figure 5(c) illustrates the next step when door D opens and no pedestrian agents will pass through door D. This phenomenon happens because the agents do not have any information regarding the doors except for the nearest door.

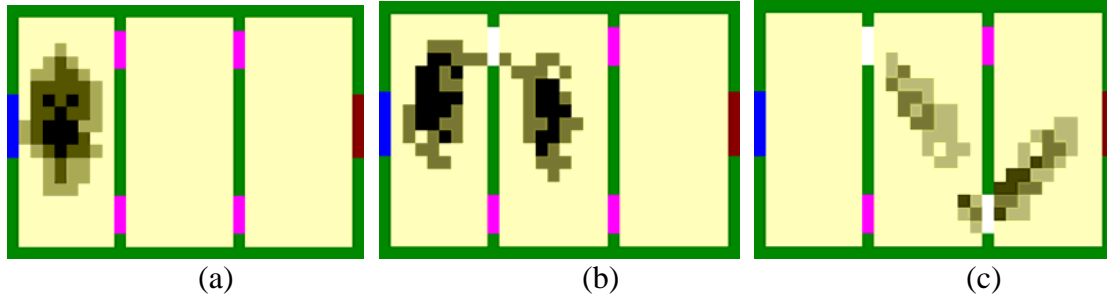


Figure 6. Running simulation in dynamic environment at full information ($\gamma = 1$)

In contrast to the second scenario, the third scenario was concerns a dynamic environment where all agents have full information regarding all routes in the infrastructure ($\gamma = 1$). Pedestrian agents do not queue in front of doors. They move around in the middle of the first room to optimize the way to the doors probably opening because they don't know which door will be opened at first as illustrated in Figure 6(a). When door A is opened, several pedestrian agents start to enter the second room and move around in the second room as shown in Figure 6(b). Then, door D is opened and all pedestrian agents move through door A and then through door D as shown in Figure 6(c).

The two extreme parameter settings on the two scenarios in dynamic environments above do not fully represent the realistic behaviour of pedestrians in the real world. More realistic pedestrian agent behaviour in front of closed and open doors shall be emphasized to produce the following behaviours:

- Pedestrians can queue in front of the door.
- Pedestrians are attracted to pass through a door where there is a queue when the door is opened, unless there is another door nearby, which is open. A very far open door will be disregarded.
- Pedestrians go to the next room if the door in front of the agent is opened and stay there waiting for the next door to be opened. This behaviour happens regardless whether there is a cell connection to the sink basin or not for the current layout configuration, as long as a connecting door ((i.e. closed door) to the sink basin is existing.

A more realistic behaviour as listed above can be reached when both a pre-computed initial navigation matrix and a re-computation of the navigation matrix for the current layout configuration exist. In other words, the value of information degree γ must be adjusted to produce behaviour as listed above. Different layout plans need different

settings of the parameter; for the layout plan with three rooms above, we found that γ range about 0.7 to 0.85 produces quite satisfying results as shown in figure 7. Some pedestrian agents still queue in front of the closed door (due to lack of information) while some of them are proceeding to the destination by a finding new route. Of course the trajectory path of pedestrian agents in a dynamic environment will not always have shortest length.

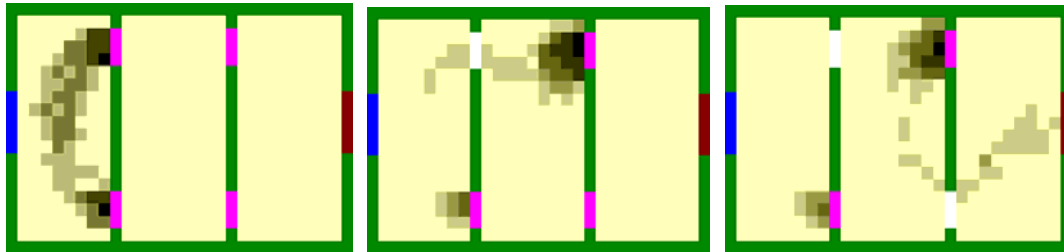


Figure 7. Running simulation in dynamic environment at $\gamma = 0.7$

CONCLUSIONS & FURTHER STUDIES

We attempted to model a pedestrian agent's wayfinding at a mesoscopic level of detail in a dynamic environment where the doors can be opened and closed at running time. Using normal pre-computation procedures, the combination of layout configuration soon goes beyond feasible computation. We generalize the navigation matrix based on a smoothing relaxation of spatial lattice permission to produce a more realistic behaviour of pedestrian agents that could queue in front of closed doors, move to the next room when the door is opened and find another door if a farther door is the one which is opened. It is quite appealing that this simple model could mimic human behaviour in navigation, in queuing and walking nearer to the shortest path and finding the way to the new open door. It was also found that when the corresponding agents' focus on the previous knowledge, they tend to follow a global pattern of previous experience.

Still there are several questions raising in this introductory approach. How to calibrate such a model spatially are still open question. Comparison of the flow performance of more complex scenarios and the effect of changing parameters such as information degree, strength of attraction and speed density relationship in other types of facilities are also subject to further studies.

ACKNOWLEDGEMENT

This research is supported by the Austrian Science Fund (FWF) through Hertha-Finberg Fellowship. Special thanks to Dietmar Bauer, Norbert Brändle and Gloria P. Gerilla who read the first draft of the paper.

REFERENCES

Blue, V.J. and Adler, J.L. (2000) Cellular Automata Microsimulation of Bidirectional Pedestrian Flows, Transportation Research Board 1678, pp.135-141.

- Helbing D. and Molnár P. (1995) Social force model for pedestrian dynamics, *Physical Review E* 51, 4282-4286
- Hershberger, J. and Suri, S. (1999). An optimal algorithm for Euclidean shortest path
- Hoogendoorn, S. P. and Bovy, P. H. L. (2004) Pedestrian route-choice and activity scheduling theory and models, *Transportation Research Part B: Methodological*, Volume 38, Issue 2 , February 2004, 169-190
- Kretz, T. and Michael Schreckenberg, M. (2006) F.A.S.T.- Floor field- and Agent-based Simulation Tool, , *Proceeding of International Symposium of Transport Simulation 2006*.
- Latombe, J. (1995). *Robot Motion Planning*. Kluwer Academic Publications, Boston.
- Lee, D. T. (1983). Visibility of a simple polygon," *Computer Vision, Graphics, and Image Processing*, vol. 22, pp. 207-221.
- Neufert (2000) *Neufert Architects Data 3rd ed*, Blackwell Publishing Company.
- Schadschneider A, 2001, "Cellular automaton approach to pedestrian, dynamics-theory", in M. Schreckenberg, S. D. Sharma (eds.) *Pedestrian and Evacuation Dynamics*, Springer, Berlin
- Teknomo, K. and Gerilla, G. P. (2005) Sensitivity Analysis and Validation of a Multi-Agents Pedestrian Model, *Journal of the Eastern Asia Society for Transportation Studies (EASTS)*, Vol 6, September 2005, p.198-213
- Winston, P. H. (1993) *Artificial Intelligence 3rd ed.*, Addison-Wesley Publishing Company, 1993, Reading, Massachusetts.
- Wolfram, S. (2002) *A New Kind of Science*, Wolfram Media, Inc.