Proceedings of the 2012 Winter Simulation Conference
C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A.M. Uhrmacher, eds

ABOUT THE PD CROWD SIMULATION FRAMEWORK

Jeroen Bijsterbosch

INCONTROL Simulation Solutions
Papendorpseweg 77
NL-3528 BJ Utrecht, THE NETHERLANDS

Wouter van Toll

University of Utrecht
Department of Information and Computing Sciences
Buys Ballot Laboratory, office 417
Princetonplein 5, De Uithof
3584 CC Utrecht, THE NETHERLANDS

Holger Pitsch
INCONTROL Simulation Solutions
Gustav-Stresemann-Ring 1
D-65189 Wiesbaden, GERMANY

ABSTRACT

As the world population is growing and urbanization increases, the focus on efficient and safe crowd management is growing. In all kinds of environments the importance of analyzing and quantifying crowd flows is acknowledged. The quality of crowd flows and particularly the safety in pedestrian environments are more important than ever before.
To support in this INCONTROL Simulation Solutions developed Pedestrian Dynamics: a brand new, state- of-the-art simulation platform to simulate large crowds in complex environments. This paper gives an insight in the used scientific techniques and its implementation details within Pedestrian Dynamics.

1 INTRODUCTION

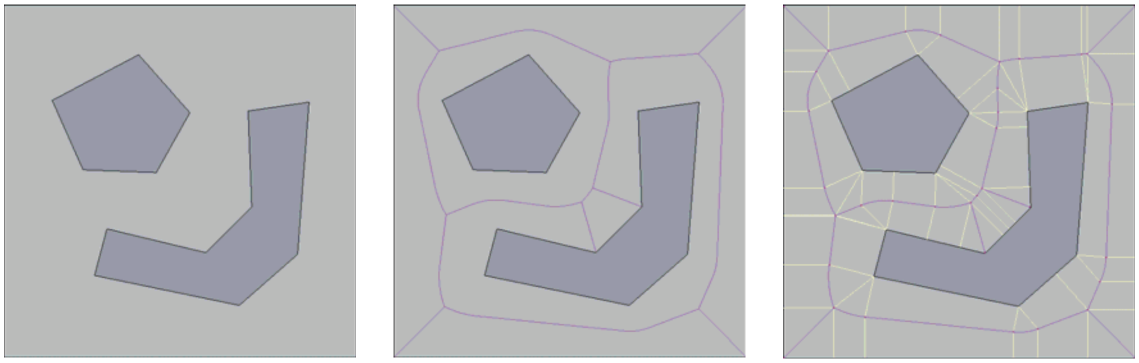
Pedestrian Dynamics (PD) can be used to model, analyse, optimize and visualize large crowds of virtual pedestrians (agents) in real-time. By importing your infrastructural design in the software, you can quickly build your own simulation model for small and large scale multi-level environments. The automated and dynamic routing and user friendly interface enables you to easily setup and conduct experiments. With the standardized output modules you can easily evaluate the performance by checking for instance densities, flows, walking times and waiting times. To achieve these results, PD uses efficient crowd simulation algorithms and software, developed together with the University of Utrecht, The Netherlands. This paper gives an introduction to the PD crowd simulation framework. Interested readers can find more details in the referenced scientific publications.

2 NAVIGATION MESH - EXPLICIT CORRIDOR MAP

During the simulation, agents should be able to efficiently find a path from their current position to any other position in the environment. A data structure that can answer these path planning questions is called a navigation mesh: a subdivision of the entire walkable space into connected polygonal areas.

One example of a navigation mesh is the Explicit Corridor Map (ECM) **[1]**; an efficient data structure describing the navigation mesh for multi-layered environments. The ECM is essentially a network (or a graph) consisting of vertices and edges. Hence, PD often refers to this data structure as the "ECM network". The edges of the ECM form the medial axis: a set of curves describing the middle of the walkable space. Figure 1a for an example environment; Figure 1b shows its medial axis.

Each ECM edge consists of nodes annotated with closest (obstacle) points, which induce a subdivision of the walkable space into polygonal areas. Hence, the closest-point annotations turn the ECM from a regular graph into a navigation mesh. Figure 1c shows the closest-point data in our running example. Observe that the yellow line segments completely subdivide the free space into unique sub-areas.



(a) Environment (b) Medial axis (c) Explicit Corridor Map

Figure 1: (a) A simple environment with two obstacles, shown in dark gray. (b) The medial axis, shown in pink, runs through the middle of the walkable space. (c) Closest-point annotations, shown in yellow, turn the medial axis into the ECM navigation mesh.

When planning a path to some goal position, an agent tries to find a route along the network's edges (i.e. along the medial axis), by using a modified A* algorithm. Thanks to the ECM's closest-point annotations, the resulting route is actually a corridor: a set of polygons and circle segments, describing the free space that the agent can use around the route. Agents can move flexibly and efficiently through a corridor (see "Path following"), and they can use the free space to avoid other agents (see "Local collision avoidance"). Figure 2 shows an example of a corridor.

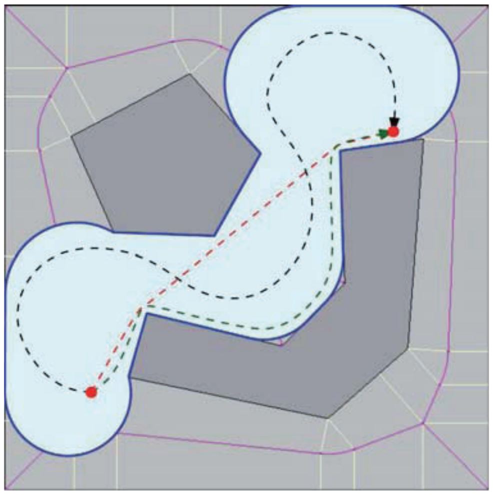


Figure 2: Planning a route between two points (shown in red) in the ECM returns a corridor, shown in blue. The agent can follow any kind of path inside this corridor, e.g. the shortest path (red), a path that stays to the right (green) or an arbitrarily shaped path (black).

2.1 Advantages of the Explicit Corridor Map

Next to its corridor flexibility, the ECM has more useful properties:

- It can be constructed quickly and automatically, given a set of layers and their obstacles. In PD, users can quickly build an arbitrary environment and then generate the routing network by pressing a single button.
- It has a small memory footprint: its size is proportional to the complexity (the total number of obstacle points) of the environment.
- It supports multi-layered environments, in which multiple two-dimensional layers are connected, e.g. through staircases or ramps **[2]**.
- It can plan paths for agents of various sizes, by using only a single data structure. Agents can decide for themselves whether or not a passage (an ECM edge) is wide enough for them to use.
- It can be annotated with more information about the environment, such as the local density (see "Density-based crowd simulation"), special edge costs (e.g. for preferring escalators over staircases), or temporary changes (e.g. staircases that become unavailable, or emergency doors that open up).

In short, the Explicit Corridor Map is an efficient and flexible navigation mesh for crowd simulation.

2.2 Route following - The Indicative Route Method

Once an agent has planned a global route to its goal position (i.e. it has found a corridor), the agent should look for a way to move through its corridor. For instance, the agent can choose to stay on the left or right side of the corridor, or to follow the shortest possible path with some preferred clearance to obstacles. Figure 2 shows a number of options.

The so-called Indicative Route Method (IRM) **[3]** is a general framework that smoothly steers an agent through a corridor while following an indicated path (the indicative route). In each step of the simulation, the agent computes a desired velocity (speed vector) that will send the agent further along its indicative route. The agent may deviate from this desired velocity, e.g. when walking around other agents, as long as it does not leave its corridor.

In PD, users can set the options for this path planning phase for each agent profile. These settings can be found in “Agent input -> Agent profile -> Route following”.

2.3 Local collision avoidance

As mentioned, path planning in corridors gives the simulated agents a lot of flexibility. Next to the described variety of indicative routes, a corridor also supports collision avoidance between agents. Collision avoidance can be time-consuming task, but it increases the simulation’s realism.

Each agent uses vision to detect which obstacles, both dynamic and static, it has to avoid. The vision is modeled as a cone-shaped field of view (FoV). The collision-avoidance algorithm in the ECM crowd simulator lets each agent chooses a velocity that is close to its desired velocity (i.e. with a small difference in direction and speed), but that prevents them from colliding with others. Similarly, the agents can be blocked by local obstacles, such as temporarily closed doors. The collision avoidance algorithm is based on the vision based model developed by Moussaïd, Helbing and Theraulaz [**5**].

Newtonian force models are still not fully consistent with empirical observations and are often hard to calibrate. Therefore, Pedestrian Dynamics uses a cognitive science approach, which is based on vision and behavioral heuristics. Guided by visual information, namely the distance of obstructions in candidate lines of sight, pedestrians apply two simple cognitive procedures to adapt their walking speeds and directions. For more detailed information see [**5**]. The model predicts the emergence of self-organization phenomena, such as the spontaneous formation of unidirectional lanes, stop-and-go waves, crowd compression, edge and wake effects and others.

In PD, users can switch agent collision avoidance on an off for the entire simulation, in “General settings -> Simulation”. Other options (e.g. the size of the field of view) can be set for each type of agent. These settings can be found in “Agent input -> Agent profile -> Local behavior”.

Note: Collision avoidance with “regular” obstacles (such as walls and buildings) can be computed very efficiently, because this information is stored in the corridors created by the ECM framework. A corridor is guaranteed to be walkable. As long as an agent stays inside its corridor, it cannot collide with stationary obstacles. The absence of finding the obstacles for collision checking is one of the reasons why ECM-based crowd simulation is efficient.

2.4 High density streams

Existing local collision avoidance algorithms don’t produce correct behavior at high densities. Concurrent flow algorithms assume too much global cooperation and lack the possibility for individual agent behavior. They are also inefficient at low densities.

Recent research **[6]** has shown that at high densities a form of inter-agent interaction is required. The basic egocentric behavior of agents is no longer sufficient. Therefore, PD extends the local collision avoidance algorithm with a streams algorithm that creates this kind of interactions at high densities.

The streams algorithm **[7]** allows agent to coordinate their movement by increasing lane formation. All agents have a desired direction in which they would like to move. Besides this desired direction, the local perceived stream is also computed. This stream represents the local flow of agents. The local stream is extracted from the nearest visible neighbors. The stream direction for a single neighbor is an interpolation between the neighbor’s direction of movement and its current relative location. By combining these interpolated directions over all relevant neighbors, the local stream can be computed.

Depending on their local density and individual preferences (some agents might not want to follow the flow), agents need to coordinate at a higher or lower degree. Therefore, an agent-based strategy interpolation is computed to determine the desired direction of the agent. This desired direction combines both coordination and personal agent preferences. The streams algorithm allows for correct individual agent behavior at both low as high densities.

.

2.5 Density-based crowd simulation

For pedestrian simulation tools such as PD, crowd density is very important. Many researchers have shown that agents generally walk at a slower pace when the local density is high. This relation can be captured in a speed-density formula. PD contains a number of commonly used formulas; users are free to change them. In literature, crowd density is often measured in persons per square meter (p/m^2), assuming that all agents have a certain (average) size. However, PD supports agents of various sizes: larger agents have a larger contribution to the crowd density. In our framework, the density is simply a value between 0 and 1 denoting how much of an area is occupied. To ensure that PD can still use density formulas from literature, the "General settings -> Simulation" window contains a setting for the "average agent area".

The Explicit Corridor Map supports route planning based on density **[4]**. Recall that each edge of the ECM denotes a set of polygonal areas through its closest-point annotations. In other words, every edge is associated to a walkable polygonal region. By keeping track of the crowd density in these regions, we approximate the density around each edge. A density formula translates this density to an *expected walking speed*, and an *expected traversal time* for the edge. The agents can use these traversal times when planning a route: this *density-based crowd simulation* lets agents avoid crowded regions, and it spreads the crowd among alternative routes in a natural-looking way.

In PD, general density-related parameters can be found in "General settings", and the routing preferences can be set for each agent type in "Agent input -> Agent profile -> Route planning".

3 COMBINATION WITH PEDESTRIAN DYNAMICS

Pedestrian Dynamics includes a software module that builds the ECM and performs crowd simulation.

In PD, the user can build an environment by defining layers and filling them with obstacles and infrastructural elements such as staircases (which are actually separate layers). The environment is then converted to a PRIX file: an XML description of the layers, their obstacles and their connections. The ECM generation software returns an ECMX file: an XML file describing the vertices and edges of the Explicit Corridor Map. Back in PD, users can visualize the ECM's edges, vertices, nodes and annotations in 2D and in 3D. When the simulation starts an ECMU file: an XML file describing specific edge properties is send to the external crowd simulation module.

During the simulation, PD generates agents and determines their goals by using the activity locations and activity routes drawn by the user. PD sends the start and goal positions of agents to the crowd simulation module, which plans the actual routes in the ECM network. In each simulation step, the module returns a new velocity for each agent.

A model in PD can also contain incidents, which trigger changes in the availability of the ECM's edges. These changes are sent, again as an ECMU file, to the crowd simulator, so that agents can respond to them in real-time.

This decoupled approach is very powerful. PD generates agents and performs their global decision-making based on activities, without requiring knowledge of the ECM from the user. In turn, the ECM simulator computes actual paths and velocities, without having to bother about the "meaning" of the environment. Combined with hardware accelerations (such as multithreading), this simulation framework can model the movement of huge crowds in real-time.

REFERENCES

[1] R. Geraerts. "Planning Short Paths with Clearance using Explicit Corridors." In *IEEE International Conference on Robotics and Automation (ICRA'10)*, pp. 1997-2004, 2010.

[2] W.G. van Toll, A.F. Cook IV, and R. Geraerts. "Navigation Meshes for Realistic Multi-Layered Environments." In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'11)*, pp. 3526-3532, 2011.

[3] I. Karamouzas, R. Geraerts, and M. Overmars. "Indicative Routes for Path Planning and Crowd Simulation". In *The Fourth International Conference on the Foundations of Digital Games (FDG'09)*, pp. 113-120, 2009.

[4] W.G. van Toll, A.F. Cook IV, and R. Geraerts. "Real-Time Density-Based Crowd Simulation." *Computer Animation and Virtual Worlds (CAVW)*, 23(1):59-69, 2012.

[**5**] M. Moussaïd, D. Helbing, G. Theraulaz. "How simple rules determine pedestrian behaviour and crowd disasters." *Proceedings of the National Academy of Science(PNAS)*, 2011.

[6] S. Lemercier, A. Jelic, R. Kulpa, J. Hua, J. Fehrenbach, P. Degond, C. Appert-Rolland, S. Donikian and J. Pettré. "Realistic following behaviors for crowd simulation." In *EUROGRAPHICS volume* 31, 2012.

[7] A. I. van Goethem. "A Stream algorithm for crowd simulation to improve crowd coordination at all densities." *Master thesis Utrecht University*, 2012. (To be published)