JÜLICH PEDESTRIAN SIMULATOR

FORSCHUNGSZENTRUM JÜLICH GMBH

User's Guide Version 0.5

Authors:

Ulrich KEMLOH
Mohcine CHRAIBI
Jun ZHANG

Stand: August 5, 2014

Contents

Listings

List of Figures

DISCLAIMER

In no event shall JuPedSim be liable to any party for direct, indirect, special, incidental, or consequential damages, including lost profits, arising out of the use of this software and its documentation, even if JuPedSim has been advised of the possibility of such damage.

JuPedSim specifically disclaims any warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The software and accompanying documentation, if any, provided hereunder is provided “as is”. JuPedSim has no obligation to provide maintenance, support, updates, enhancements, or modifications.

Forschungszentrum Jülich GmbH makes no warranty, expressed or implied, to users of JuPedSim, and accepts no responsibility for its use. Users of JuPedSim assume sole responsibility for determining the appropriateness of its use; and for any actions taken or not taken as a result of analyses performed using this tool.

Users are warned that JuPedSim is intended for **academic** use only. This tool is an implementation of several computer models that may or may not have predictive capability when applied to a specific set of factual circumstances. Lack of accurate predictions by the models could lead to erroneous conclusions with regard of life safety.

Chapter 1

JuPedSim

The Jülich Pedestrian Simulator (JuPedSim) is a tool for simulating and analysing pedestrians motion. It consists (at the moment) of three modules which are loosely coupled and can be used independently. These are:

1. **JPScore**: the core module computing the trajectories
2. **JPSvis**: tool for visualising the trajectories
3. **JPSreport**: tool for analysing the trajectories and performing measurements e.g. density, velocity, flow.

The description and use of each module is presented in the next chapters. The code is written in C++ and all modules can be compiled using *CMake* and a C++ compiler that supports the latest c++11 standard. For convenience binaries for Windows and Linux are provided.

Binaries:
The binaries are available from https://github.com/JuPedSim/JuPedSim/releases

Sources:
JuPedSim is developed at the Forschungszentrum Jülich in Germany and the bleeding edge code is in their intern git repository. At the moment only specific tags (releases) are pushed to github at https://github.com/JuPedSim/JuPedSim.

For the bleeding edge code, please contact info@jupedsim.org or have a look at http://cst.version.fz-juelich.de/ .

Showcase:
Some videos are available on our YouTube channel: https://youtube.com/user/JuPedSim

Chapter 2

Running a simulation

Perform the following steps to run a simulation from scratch:

1. Download the latest version for your architecture from https://github.com/JuPedSim/JuPedSim/releases
2. Open one of the demos folder present in the unpacked archive. There are at least three files:
   - project file: usually ends with "_ini.xml"
   - geometry file: usually ends with "_geo.xml"
   - trajectory file: usually ends with "_traj.xml"
3. Drag and Drop the trajectories file on TraVisTo.exe (Windows) TraVisTo (Linux) if you want to visualize the pre-computed trajectories.
4. Drag and Drop the project file on jpscore.exe (Windows) jpscore (Linux) in the case you want to compute the trajectories first.
5. Make some modifications to the files and enjoy.

Note In order to prevent sporadic crashes make sure to validate all your input files against the supplied definition files

- *jps_geometry.xsd* for the geometry
- *jps_ini_core.xsd* for the simulation
- *jps_routing.xsd* for the files containing additional navigation lines.

Chapter 3

JPScore

The core module for performing the simulation (i.e. computing the trajectories) is initialized with a project file containing all the necessary parameters for the simulation. All input files are XML-based. All distances are in metres (m), all times are in seconds (s)and all speeds are in metres per seconds (ms^{-1}). In the following JPSCORE denotes the executable.

3.1 Project file (ini)

A program call looks like this:

```
> jpscore --inifile=ini.xml
```

The same result is achieved by dragging and dropping the project file on the executable. The typical content of a project file (ini.xml) is described in Listing 3.1.

Listing 3.1: Structure of a project file

```
1
2  <?xml version="1.0" encoding="UTF-8" ?>
3
4  <JuPedSim project="JPS-Project" version="0.5"
5  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
6  xsi:noNamespaceSchemaLocation="http://134.94.2.137/jps_ini_core.xsd">
7
8      <!-- header: seed , geometry, output format -->
9
10     <!-- traffic information: e.g closed doors or smoked rooms -->
11     <traffic_constraints>  </traffic_constraints>
12
13     <!-- goals outside the geometry-->
14     <routing>  <goals/>  </routing>
15
16     <!--pedestrians information and distribution -->
```

```
17      <agents operational_model_id="2">
18          <agents_distribution></agents_distribution>
19      </agents>
20
21      <!-- operational models -->
22      <operational_models>
23          <model operational_model_id="1" description="gcfm" />
24      </operational_models>
25
26      <route_choice_models>
27          <router router_id="1" description="global_shortest" />
28      </route_choice_models>
29
30  </JuPedSim>
```

3.1.1 Header

The header comprises the following elements:

- `<seed>12542</seed>`

  The seed used for the random number generator. If missing the number of seconds elapsed since Jan 1, 1970 (i.e. *time(NULL)*) is used.

- `<max_sim_time>200</max_sim_time>`

  The maximal simulation time in seconds. The default value is 900 seconds. The simulation will end after this time.

- `<numCPU>4</numCPU>`

  The number of cores to be used by OpenMP.

- ```
  <trajectories format="xml-plain"  fps="8">
  <file location="corner_traj.xml" />
  <!--<socket hostname="127.0.0.1" port="8989"/> -->
  </trajectories>
  ```

  format A detailed description of the formats are presented in the next section. The options are:

  - **xml-plain**: the default output format in XML. It can lead to large files.
  - **plain**: simple text format
  ```

fps defines the frame rate per second for the trajectories. Note that the step size for the simulation usually varies from 0.001 to 0.01 due to stability issues.

file location defines the location of the trajectories. All paths are relative to the location of the project file.

Streaming Optionally to a file location a network address (ip/hostname and port) for streaming the results can also been provided.

- `<geometry>corner_geo.xml</geometry>`

  The location of the geometry file. All files locations are relative to the actual location of the project file.

3.1.2 Traffic constraints

This section defines constraints related to the traffic. At the moment doors can be open or close.

Listing 3.2: Sample traffic constraints section

```
<traffic_constraints>
        <!-- doors states are: close or open -->
        <doors>
            <door trans_id="4" caption="Main-gate" state="open" />
            <door trans_id="6" caption="Rear-gate" state="close" />
        </doors>
    </traffic_constraints>
```

trans_id unique ID of that door as defined in the geometry file (see section 3.2).

caption optional parameter defining the caption of the door.

state current state of the door. Possible values are **close** or **open**.

3.1.3 Goal definitions (routing)

Additional goals might be defined *outside* the geometry. They should not overlap with any walls or be located in a room. The best practice is to position them near

to exits. Goals are defined with close polygons i.e. the last vertex is equal to the first one. A sample goal definition is presented in Listing 3.3.

Listing 3.3: Sample goals definition

```
<routing>
        <goals>
            <goal id="0" final="false" caption="goal 1">
                <polygon>
                    <vertex px="-5.0" py="-5.0" />
                    <vertex px="-5.0" py="-2.0" />
                    <vertex px="-3.0" py="-2.0" />
                    <vertex px="-3.0" py="-5.0" />
                    <vertex px="-5.0" py="-5.0" />
                </polygon>
            </goal>
            <goal id="1" final="false" caption="goal 2">
                <polygon>
                    <vertex px="15.0" py="-5.0" />
                    <vertex px="17.0" py="-5.0" />
                    <vertex px="17.0" py="-7.0" />
                    <vertex px="15.0" py="-7.0" />
                    <vertex px="15.0" py="-5.0" />
                </polygon>
            </goal>
        </goals>
</routing>
```

3.1.4 Agents definition

This section defines different groups of pedestrians with individual characteristics. An example is given in Listing 3.4.

Listing 3.4: Sample agents parameter section

```
<agents operational_model_id="2">
    <agents_distribution>

        <group group_id="1" agent_parameter_id="1" room_id="0" subroom_id="0"
            number="10" />

        <group group_id="2" agent_parameter_id="2" room_id="0" subroom_id="1"
            number="20"     goal_id="1" router_id="2" patience="5"/>

        <group group_id="3" agent_parameter_id="1" room_id="0" number="200"
            goal_id="-1" router_id="2" patience="50"/>

        <group group_id="4" agent_parameter_id="1" room_id="0" number="1" start_x
            ="25.3" start_y="365.90"/>

        <group group_id="5" agent_parameter_id="1" room_id="0" subroom_id="1"
            number="30" patience="44" x_min="6.52" x_max="41" y_min="6.52" y_max
```

```
                =" 49 " />
13
14      </ agents_distribution>
15  </ agents>
```

operational_model_id mandatory parameter defining the operational model to be used. Values can be 1 (gcfm) or 2 (gompertz)

group_id mandatory parameter defining the unique id of the group.

agent_parameter_id mandatory parameter defining the agents characteristics. This is used to simulate children or elderly for instance. See subsection 3.1.5 for more information.

room_id mandatory parameter defining the room where the agents should be randomly distributed.

number mandatory parameter defining the number of agents to distribute.

subroom_id defines the id of the subroom (see section 3.2) where the agents should be distributed. If omitted then the agents are homogeneously distributed in the room. See the geometry section.

goal_id one of the *id* as defined in routing subsection 3.1.3. If omitted or "-1" then the shortest path to the outside is preferred by the agent.

router_id defines the route choice model to used. Those models are defined in the section subsection 3.1.6

patience in seconds influences the route choice behavior when using the quickest path router. It basically defines how long a pedestrian stays in jams before attempting a re-routing.

start_x, start_y define the initial coordinate of the agents. This might be useful for testing. Note that these coordinates are ignored if *number* is different from 1.

x_min, x_max y_min, y_max define additional constraints where to distribute the agents. Not all variable must be defined. For instance [x_min="0", y_min="0"] will only distribute agents in the intersection of the first quadrant and the specified room/subroom.

3.1.5 Operative models

JuPedSim comes with two implementation of force-based models:

- Generalized Centrifugal Force Model (GCFM), with the id=1
- Gompertz model, with the id=2

Each model definition has at least two sections:

- model_parameters. Here are the definition of the parameters specific to the model, for instance the force range.
- agent_parameters. Here are the definition of the parameter specific to the agents, for instance their size or the desired velocity, as far as accounted by the model. This allows to attribute different characteristics to different groups, for instance children or elderly.

GCFM: operational_model_id=1

For the definition of the GCFM the user is refereed to [2].

Listing 3.5: Sample definition of the GCFM

```
<operational_models>
    <model operational_model_id="1" description="gcfm">
        <model_parameters>
            <solver>euler</solver>
            <stepsize>0.01</stepsize>
            <exitCrossingStrategy>4</exitCrossingStrategy>
            <linkedcells enabled="true" cell_size="2.2" />
            <force_ped nu="0.3" dist_max="3" disteff_max="2" interpolation_width=
                "0.1" />
            <force_wall nu="0.2" dist_max="3" disteff_max="2" interpolation_width
                ="0.1" />
        </model_parameters>
        <agent_parameters agent_parameter_id="1">
            <v0 mu="0.5" sigma="0.0" />
            <bmax mu="0.25" sigma="0.001" />
            <bmin mu="0.20" sigma="0.001" />
            <amin mu="0.18" sigma="0.001" />
            <tau mu="0.5" sigma="0.001" />
            <atau mu="0.5" sigma="0.001" />
        </agent_parameters>
```

```
19          <agent_parameters agent_parameter_id="2">
20              <v0 mu="0.5" sigma="0.0" />
21              <bmax mu="0.25" sigma="0.001" />
22              <bmin mu="0.20" sigma="0.001" />
23              <amin mu="0.18" sigma="0.001" />
24              <tau mu="0.5" sigma="0.001" />
25              <atau mu="0.5" sigma="0.001" />
26          </agent_parameters>
27      </model>
28  </operational_models>
```

operational_model_id unique identifier of this model. The id is referred in the agent definition (see subsection 3.1.4).

agent_parameter_id unique identifier for this group. The id is referred in the agent distribution definition (see subsection 3.1.4).

solver the Euler scheme is used to solve the differential equation resulting from the agents equation of motion.

stepsize integration step size in seconds. Be careful not to used a large value otherwise the system might become unstable. The recommended value 10^{-3} s.

exitCrossingStrategy defines how a pedestrian crosses a line $L = [P_1, P_2]$. Possible values are:

- **1**: The direction of the pedestrian is towards the middle of L ($\frac{P_1+P2}{2}$)
- **2**: The direction is given by the nearest point on L to the position of the pedestrian.
- **3**: Same as 2, only the line L is shorten on both edges by 20 cm.
- **4**: If the nearest point of the pedestrian on the segment line L is outside the segment, then chose the middle point as target. Otherwise the nearest point is chosen.
- **5**: This strategy is still beta. It assumes that the simulation scenario as no loops or U-shaped corridors. Pedestrians, are steered towards the exit, even if it is not within their visibility range.

Most of the aforementioned strategies are discussed in [1].

linkedcell The linked-cells reduce the runtime by optimizing the neighbourhood query of pedestrians. Chose *false* or a large *cell_size* if you want to consider all pedestrians as neighbours. This corresponds to a brute force approach summing over all other pedestrians, while calculating the repulsive forces. Depending on the number of pedestrians, the simulation can take some time [5].

v0 $\sim \mathcal{N}(\mu, \sigma)$.
The desired velocity v_0 is Gaussian distributed between $[\mu - \sigma, \mu + \sigma]$ with the parameters μ and σ. μ defaults to 1.24

bmax, bmin $\sim \mathcal{N}(\mu, \sigma)$.
The original published version of the model for the semi-axis b has two fundamental problems (See [2]):

- pedestrians cannot have a zero desired speed.
- the linear dependency of the speed is, in fact a simplification, but is not supported by empirical data.

Following modifications are implemented in JuPedSim. We consider the following relation $f(v) = a \exp(bv)$ for the speed v and fit the parameters a and b according to the empirical data of the 1D experiment [6] (See Figure 3.1).

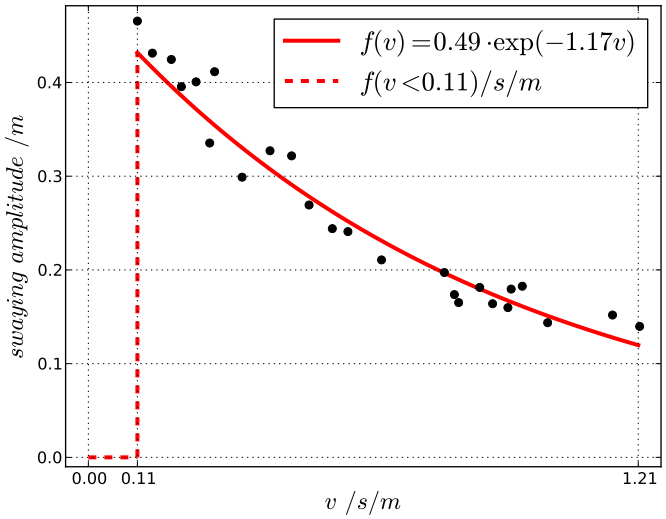


Figure 3.1: Fitting the amplitude of the swaying with respect to the speed of pedestrians.

Considering the semi-axis in the shoulder direction as the space required taken by one pedestrian while moving (including his lateral swaying) and assuming that the shoulder axis is $b_{\text{shoulder}} = 40$ cm, we set:

$$b(v) = \frac{1}{2}\Big(b_{\text{shoulder}} + f(v)\Big), \tag{3.1}$$

with $a = 0.49\,/m$, $b = -1.17/m/s$.

amin, atau $\sim \mathcal{N}(\mu, \sigma)$.
the length of the semi-axis in the direction of motion is defined by:

$$a = a_{\text{min}} + a_\tau v.$$

The parameter a_{min} and a_τ are Gaussian distributed.

force_ped, force_wall parameter for the repulsive force among pedestrians and with walls. See [2] for description of the parameter.

Gompertz model: operational_model_id=2

This model is unpublished and is fully described in Appendix A. It is based on a continuous "physical" force. Many parameters defined for the GCFM are also valid for the Gompertz model. Both models differ only in the definition of the repulsive force (see Equation A.1 and Equation A.3).

Listing 3.6: Sample definition of the Gompertz model

```
<operational_models>
    <model operational_model_id="2" description="gompertz">
        <model_parameters>
            <solver>euler</solver>
            <stepsize>0.01</stepsize>
            <exitCrossingStrategy>3</exitCrossingStrategy>
            <linkedcells enabled="true" cell_size="2.2" />
            <v0 mu="0.5" sigma="0.0" />
            <bmax mu="0.25" sigma="0.01" />
            <bmin mu="0.20" sigma="0.01" />
            <amin mu="0.22" sigma="0.02" />
            <tau mu="0.5" sigma="0.00" />
            <atau mu="0.1" sigma="0.01" />
            <force_ped nu="3" b="0.25" c="3.0"/>
            <force_wall nu="10" b="0.70" c="3.0"/>
        </model_parameters>
        <agent_parameters agent_parameter_id="1">
            <v0 mu="0.5" sigma="0.0" />
```

```
19              <bmax mu="0.25" sigma="0.001" />
20              <bmin mu="0.20" sigma="0.001" />
21              <amin mu="0.18" sigma="0.001" />
22              <tau mu="0.5" sigma="0.001" />
23              <atau mu="0.5" sigma="0.001" />
24          </agent_parameters>
25          <agent_parameters agent_parameter_id="2">
26              <v0 mu="0" sigma="0.0" />
27              <bmax mu="0.25" sigma="0.001" />
28              <bmin mu="0.20" sigma="0.001" />
29              <amin mu="0.18" sigma="0.001" />
30              <tau mu="0.5" sigma="0.001" />
31              <atau mu="0.5" sigma="0.001" />
32          </agent_parameters>
33      </model>
34  </operational_models>
```

force_ped, force_wall parameter for the repulsive force among pedestrians and with walls. See Appendix A for a detailed description of the parameters.

- nu strength of the force
- b displacement along the x-axis $\equiv$ cut off radius
- c growth rate (y scaling).

3.1.6 Route choice models

Two route choice models are available at the moment. There are a lot more coming up with the next version. A sample definition is given in Listing 3.7.

Global shortest path

The default route choice algorithm is the global shortest path. At the beginning of the simulation, An evacuation network is generated based on the information present in the geometry file. This are mainly the location of the exits (and other navigation lines) and the states of the exits (closed/opened). The Dijkstra algorithm is then used to compute the shortest paths to the defined destinations. This algorithm is static and the agents do not adjust their routes, based for instance, on traffic conditions.

Quickest path

In this route choice algorithm, the agents are re-routed based on actual traffic conditions. The quickest path approach, which is based on the observation of the

environment by the agents, is not sensitive to initial distribution of pedestrians or special topologies like symmetric exits, making it a general purpose algorithm.

The patience time can be used to control this behavior. This defaults to the global shortest path when the patience time of the agents are very large. Detailed information about the models are presented in: [4]

Listing 3.7: Sample route choice models

```
<route_choice_models>
    <router router_id="1" description="quickest">
        <parameters>
            <navigation_lines file="routing.xml" />
        </parameters>
    </router>

    <router router_id="2" description="global_shortest">
        <parameters>
             <navigation_lines file="routing.xml" />
        </parameters>
    </router>
</route_choice_models>
```

In the case the geometry is not made of convex units (subrooms), additional navigations lines might be necessary to achieve a fluent motion for the agents, e.g at corners. An example file is given in Listing 3.8.

Listing 3.8: Sample navigation lines file

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<routing version="0.5">
    <Hlines>
        <Hline id="2" room_id="0" subroom_id="1">
            <vertex px="15.0" py="4.0" />
            <vertex px="17.0" py="6.0" />
        </Hline>
        <Hline id="3" room_id="0" subroom_id="1">
            <vertex px="15.0" py="4.0" />
            <vertex px="15.0" py="6.0" />
        </Hline>
        <Hline id="4" room_id="0" subroom_id="1">
            <vertex px="15.0" py="4.0" />
            <vertex px="17.0" py="4.0" />
        </Hline>
    </Hlines>
</routing>
```

- **id** mandatory unique identifier for this navigation line.

- **room_id** the room containing the navigation line.
- **subroom_id** the subroom containing the navigation line.
- **vertex** the coordinates of the navigation line.

3.1.7 Sample project file

A complete example of a project file for the simulation is given in Listing 3.9. More examples are foind in the demos folder which comes along side with the sources.

Listing 3.9: Sample project file

```
<?xml version="1.0" encoding="UTF-8" ?>

<JuPedSim project="JPS-Project" version="0.5"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="../../xsd/jps_ini_core.xsd">

    <!-- seed used for initialising random generator -->
    <seed>12542</seed>
    <!-- geometry file -->
    <geometry>bottleneck_geo.xml</geometry>
    <!-- traectories file and format -->
    <trajectories format="xml-plain"  fps="8">
        <file location="bottleneck_traj.xml" />
    </trajectories>
    <!-- where to store the logs -->
    <logfile>log.txt</logfile>

    <!-- traffic information: e.g closed doors or smoked rooms -->
    <traffic_constraints>
        <!-- doors states are: close or open -->
        <doors>
            <door trans_id="2" caption="main_exit" state="open" />
        </doors>
    </traffic_constraints>

    <routing>
        <goals>
            <goal id="0" caption="goal 1">
                <polygon>
                    <vertex px="70" py="101" />
                    <vertex px="70" py="103" />
                    <vertex px="75" py="103" />
                    <vertex px="75" py="101" />
                    <vertex px="70" py="101" />
                </polygon>
            </goal>
        </goals>
    </routing>

    <!--persons information and distribution -->
    <agents operational_model_id="1">
        <agents_distribution>
```

```
43            <group group_id="1" agent_parameter_id="1" room_id="1" subroom_id="0"
                    number="40" goal_id="0" router_id="1" />
44        </agents_distribution>
45    </agents>
46
47     <!-- These parameters may be overwritten -->
48     <operational_models>
49        <model operational_model_id="1" description="gcfm">
50          <model_parameters>
51            <solver>euler</solver>
52            <stepsize>0.01</stepsize>
53            <exitCrossingStrategy>4</exitCrossingStrategy>
54            <linkedcells enabled="true" cell_size="2.2" />
55            <force_ped nu="0.3" dist_max="3" disteff_max="2" interpolation_width=
                    "0.1" />
56            <force_wall nu="0.2" dist_max="3" disteff_max="2" interpolation_width
                    ="0.1" />
57          </model_parameters>
58          <agent_parameters agent_parameter_id="1">
59            <v0 mu="0.5" sigma="0.0" />
60            <bmax mu="0.25" sigma="0.001" />
61            <bmin mu="0.20" sigma="0.001" />
62            <amin mu="0.18" sigma="0.001" />
63            <tau mu="0.5" sigma="0.001" />
64            <atau mu="0.5" sigma="0.001" />
65          </agent_parameters>
66          <agent_parameters agent_parameter_id="2">
67            <v0 mu="0.5" sigma="0.0" />
68            <bmax mu="0.25" sigma="0.001" />
69            <bmin mu="0.20" sigma="0.001" />
70            <amin mu="0.18" sigma="0.001" />
71            <tau mu="0.5" sigma="0.001" />
72            <atau mu="0.5" sigma="0.001" />
73          </agent_parameters>
74        </model>
75
76        <model operational_model_id="2" description="gompertz">
77          <model_parameters>
78                <solver>euler</solver>
79                <stepsize>0.01</stepsize>
80                <exitCrossingStrategy>3</exitCrossingStrategy>
81                <linkedcells enabled="true" cell_size="2.2" />
82                <force_ped nu="3"  b="0.25" c="3.0"/>
83                <force_wall nu="10" b="0.70" c="3.0"/>
84          </model_parameters>
85          <agent_parameters agent_parameter_id="1">
86                <v0 mu="0.5" sigma="0.0" />
87                <bmax mu="0.25" sigma="0.001" />
88                <bmin mu="0.20" sigma="0.001" />
89                <amin mu="0.18" sigma="0.001" />
90                <tau mu="0.5" sigma="0.001" />
91                <atau mu="0.5" sigma="0.001" />
92          </agent_parameters>
93          <agent_parameters agent_parameter_id="2">
94                <v0 mu="0" sigma="0.0" />
95                <bmax mu="0.25" sigma="0.001" />
96                <bmin mu="0.20" sigma="0.001" />
97                <amin mu="0.18" sigma="0.001" />
98                <tau mu="0.5" sigma="0.001" />
99                <atau mu="0.5" sigma="0.001" />
100         </agent_parameters>
101       </model>
```

```
102     </operational_models>
103
104     <route_choice_models>
105         <router router_id="1" description="global_shortest">
106             <parameters>
107          <navigation_lines file="routing.xml" />
108             </parameters>
109         </router>
110     </route_choice_models>
111
112 </JuPedSim>
```

3.2 Geometry

The general structure of the geometry file is shown in Figure 3.2. A sample geometry code is given in Listing 3.10 and the corresponding visualisation in Figure 3.3.

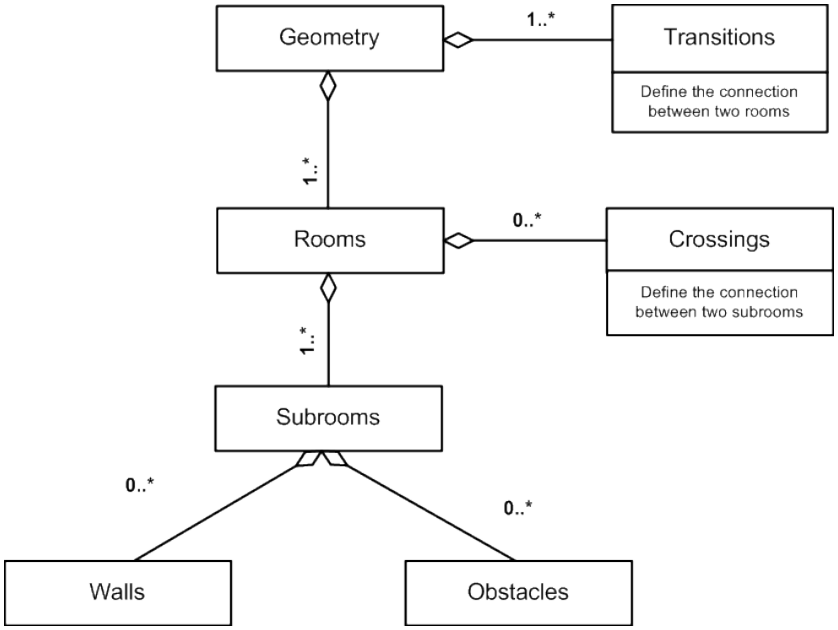


Figure 3.2: Structure of the geometry file.

Listing 3.10: Sample geometry file with one obstacle

```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <geometry version="0.5" caption="corner"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
xsi:noNamespaceSchemaLocation="http://xsd.jupedsim.org/jps_geometry.xsd">
    <rooms>
        <room id="0" caption="hall" >
            <subroom id="0"  class="subroom">
                <polygon caption="wall">
                    <vertex px="10.0" py="4.0"/>
                    <vertex px="10.0" py="0.0"/>
                    <vertex px="0.0" py="0.0"/>
                    <vertex px="0.0" py="10.0"/>
                    <vertex px="10.0" py="10.0"/>
                    <vertex px="10.0" py="6.0"/>
                </polygon>
            </subroom>
            <subroom id="1"  class="corridor">
                <polygon caption="wall">
                    <vertex px="10.0" py="6.0"/>
                    <vertex px="17.0" py="6.0"/>
                    <vertex px="17.0" py="-5.0"/>
                </polygon>
                <polygon caption="wall">
                    <vertex px="15.0" py="-5.0"/>
                    <vertex px="15.0" py="4.0"/>
                    <vertex px="10.0" py="4.0"/>
                </polygon>

                <obstacle id="0" caption="table" height="1.0" >
                    <polygon>
                        <vertex px="12.0" py="6.0"/>
                        <vertex px="13.0" py="6.0"/>
                        <vertex px="13.0" py="5.5"/>
                        <vertex px="12.0" py="5.5"/>
                        <vertex px="12.0" py="6.0"/>
                    </polygon>
                </obstacle>
            </subroom>
            <crossings>
                <!-- virtual exits between subrooms -->
                <crossing id="0" subroom1_id="0" subroom2_id="1">
                    <vertex px="10.0" py="6.0"/>
                    <vertex px="10.0" py="4.0"/>
                </crossing>
            </crossings>
        </room>
    </rooms>

    <transitions>
        <!-- exits like crossings but between rooms or to outside (room with
             index
        = -1) -->
        <transition id="1" caption="main exit" type="emergency"
            room1_id="0" subroom1_id="1" room2_id="-1" subroom2_id="-1">
            <vertex px="15.0" py="-5.0"/>
            <vertex px="17.0" py="-5.0"/>
        </transition>
    </transitions>
</geometry>
```

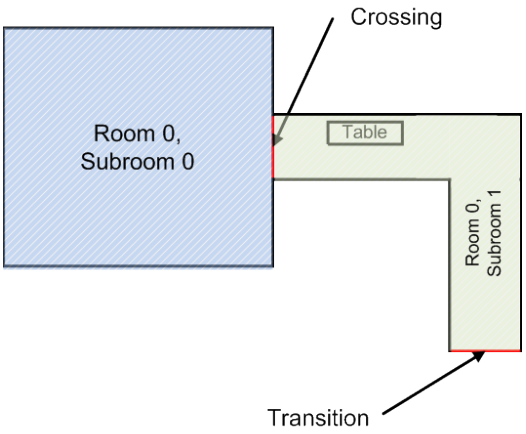


Figure 3.3: Sample geometry with one room, two subrooms and one obstacle. The obstacle should not overlap with any wall of the geometry.

3.2.1 Rooms

The geometry contains at least one *room* and one *transition*. Each room has a unique *id* and and optional caption. Two rooms are separated by either walls or transitions.

3.2.2 Subrooms

The subrooms define the navigation mesh, i.e the walkable areas in the geometry. Each subroom is bounded by at least 1 crossing. A sample subroom definition is described in Listing 3.11. To ease navigation, it is recommended to always use convex subrooms. In the case the subroom is not convex, additional navigation lines might be required, see Listing 3.8.

- **id** mandatory parameter, also referred by crossings.
- **class** optional parameter defining the class of this subroom. At the moment two classes are defined:
  - **floor**
  - **stairs** takes an additionally

    ```
    <up px="-5.0" py="2" />
    <down px="0.0" py="2" />
    ```

    used in the visualisation, for marking the highest and the lowest point.

- **A_x, B_y, C_z** optional plane equation of the subroom. It allows the construction of a 3D environment (See Appendix B).
- **polygon** describes the walls as a sequence of vertex.

Listing 3.11: Sample stair subroom

```
<subroom id="1" class="stair" A_x="-1.2" B_y="0" C_z="0">
        <polygon caption="wall">
            <vertex px="0.0" py="1.0" />
            <vertex px="-5.0" py="1.0" />
        </polygon>
        <polygon caption="wall">
            <vertex px="0.0" py="3.0" />
            <vertex px="-5.0" py="3.0" />
        </polygon>
        <up px="-5.0" py="2" />
        <down px="0.0" py="2" />
</subroom>
```

3.2.3 Transitions

A *transition* defines the connection between two rooms and is basically a door. They can be close or open in the configuration in the project file. See subsection 3.1.2.

Listing 3.12: Sample transition between two rooms

```
<!-- exits between rooms or to outside (room with index = -1) -->
    <transition id="1" caption="main exit" type="emergency"
        room1_id="0" subroom1_id="1" room2_id="-1" subroom2_id="-1">
        <vertex px="15.0" py="-5.0"/>
        <vertex px="17.0" py="-5.0"/>
    </transition>
```

- **id**, mandatory unique identifier for this door. The id is also used to close or open the door in the traffic constraints section of the project file. See subsection 3.1.2.
- **caption**, optional and displayed in the visualisation.
- **type**, optional. The parameter is not internally used.
- **room1_id**, the first room sharing this transition. The order is not important.

- **subroom1_id**, the first subroom located in room_1.
- **room2_id**, the second room sharing this transition. The order is not important. If there is no second room (meaning this transition is connected to the outside), then use **-1**.
- **subroom2_id**, the second subroom sharing this transition. The order is not important. If there is no second subroom (meaning this transition is connected to the outside), then use **-1**.
- **vertex**, the geometry of the transition as a straight line.

3.2.4 Crossings

A *crossing* defines the connection between two subrooms inside the same room. Unlike transition, they cannot be close or open. They are always open.

Listing 3.13: Sample crossing between two subrooms

```
<!-- virtual exits between subrooms -->
<crossing id="0" subroom1_id="0" subroom2_id="1">
    <vertex px="10.0" py="6.0"/>
    <vertex px="10.0" py="4.0"/>
</crossing>
```

- **id**, mandatory unique identifier for this crossing. The id is also used to close or open the door in the traffic constraints section of the project file. See subsection 3.1.2.
- **subroom1_id**, the first subroom located in room_1.
- **subroom2_id**, the second subroom sharing this transition. The order is not important. If there is no second subroom (meaning this transition is connected to the outside), then use **-1**.
- **vertex**, the geometry of the transition as a straight line.

3.2.5 Obstacles

One or more obstacles can also be defined within a subroom. They should completely reside inside the subroom. Especially there should not any intersections with other geometry elements (walls, crossings or transitions).

Listing 3.14: Sample obstacle in a subroom

```
<obstacle id="0" caption="table" height="1.0" >
    <polygon>
        <vertex px="12.0" py="6.0"/>
        <vertex px="13.0" py="6.0"/>
        <vertex px="13.0" py="5.5"/>
        <vertex px="12.0" py="5.5"/>
        <vertex px="12.0" py="6.0"/>
    </polygon>
</obstacle>
```

- **id**, mandatory unique identifier for this obstacle.
- **caption**, used in the visualisation.
- **height**, optional parameter, not used at the moment
- **polygon**, describing the obstacle as a sequence of vertex.

3.3 Trajectories

The results of the simulation are written to files or streamed to a network socket. Possible formats are:

- **xml-plain** which is the default xml format
- **plain** a flat format (just numbers)

Note that if you are using the streaming mode, the format is forced to xml-plain.

3.3.1 xml-plain

The file has three main sections: header, geometry and frames.

Header

```
<header version = "0.5">
    <agents>1</agents>
    <frameRate>8</frameRate>
</header>
```

- **agents:** The total number of agents at the beginning of the simulation.
- **frameRate:** The framerate. Divide the total number of frames by the framerate to obtain the overall evacuation time.

Geometry

The geometry can be completely embedded within the trajectories or a reference to a file can be supplied.

```
<geometry>
    <file location="corridor_geometry.xml"/>
</geometry>
```

Frames

```
<frame ID="0">
<agent ID="1"   x="660.00"   y="333.00"   z="30.00"
     rA="17.94"   rB="24.94"   eO="-168.61" eC="0"/>
</frame>

<frame ID="1">
<agent ID="1"   x="658.20"   y="332.86"   z="30.00"
rA="31.29"   rB="23.87"   eO="-175.41" eC="54"/>
</frame>
```

- **ID** mandatory, the id of the pedestrians starting with 1.
- **x, y, z** mandatory, the position of the agent.
- **rA, rB** The shape which is defined by a circle (ellipse) drawn around a human-like figure. “radiusA” and “radiusB” are respectively the semi major axis and the semi minor axis of the ellipse, if the modelled pedestrians’ shape is an ellipse. Else if it is a circle those values should be equal to the radius of the circle.
- **eO, eC** are the “ellipseOrientation” and the ”ellipseColor”. “ellipseOrientation” is the angle between the major axis and the X-axis (zero for circle). A color can also be provided, for example for displaying change in velocity. The colours are in the range [0=red, 255=green] and define the rapport between the desired velocity (V_0) and the instantaneous velocity.

Sample trajectory file

Listing 3.15: Sample trajectory file in XML format

```
<?xml version="1.0" encoding="UTF-8"?>

<trajectories>
    <header version = "0.5">
        <agents>1</agents>
        <frameRate>8</frameRate>
    </header>

    <geometry>
        <file location="corridor_geometry.xml"/>
    </geometry>

    <frame ID="0">
        <agent ID="1"   xPos="660.00"   yPos="333.00"   zPos="30.00"
        radiusA="17.94" radiusB="24.94" ellipseOrientation="-168.61"
        ellipseColor="0"/>
    </frame>

    <frame ID="1">
        <agent ID="1"   xPos="658.20"   yPos="332.86"   zPos="30.00"
    radiusA="31.29" radiusB="23.87" ellipseOrientation="-175.41"
    ellipseColor="54"/>
    </frame>
</trajectories>
```

3.3.2 plain

A sample trajectory in the plain format is presented in Listing 3.16. The second line is always the *framerate*.

Listing 3.16: Sample trajectories file in plain text format

```
                #description: my super simulation
                #framerate: 16
                #geometry: /home/sim/corridor.xml
                #ID: the agent ID
                #FR: the current frame
                #X,Y,Z: the agents coordinates in metres

                #ID FR  X       Y       Z
                1   0   28.21   131.57  0.00
                2   0   38.41   133.42  0.00
                1   1   28.21   131.57  0.00
                2   1   38.41   133.42  0.00
                1   2   28.24   131.57  0.00
                2   2   38.44   133.42  0.00
                1   3   28.29   131.57  0.00
                2   3   38.49   133.42  0.00
```

```
                1    4    28.36    131.57    0.00
                2    4    38.56    133.42    0.00
                1    5    28.44    131.57    0.00
                2    5    38.64    133.42    0.00
                1    6    28.54    131.57    0.00
                2    6    38.74    133.42    0.00
                1    7    28.65    131.57    0.00
                2    7    38.85    133.42    0.00
                1    8    28.77    131.57    0.00
```

Chapter 4

Verification and validation

In order to control the quality of this software we are setting up a list of validation tests. Some of them are inspired from the Fire Dynamic Simulator (FDS) [3].

4.1 Setting up a test

Several validation and verification tests for JuPedSim are defined in the following section. In order to make the nightly builds run automatically, consider the following steps, before adding new tests. This procedure is also recommended to make simulations with several inifiles e.g. different seeds.

1. Create in *jpscore/Utest* a new directory with a descriptive name. For example: *Utest/test_case1*

2. Put in that directory an ini-file (refereed to as "master inifile") and all the relevant files for a simulation, e.g. geometry file, routing file etc. In the master inifile you can use python syntax to give different values for the following tags:

    - **max_sim_time**: Max time of simulation
    - **seed**: You should always change this one to get credible results
    - **geometry**: Give a directory name containing different xml-files. In case you have only one file, just specify its name.
    - **numCPU**: (Optional) Number of CPUs to be used in the simulation. If not given (and openmp is activated), the maximum of cores in the machine is used. Otherwise numCPU=1.
    - **number**: Number of agents
    - **exitCrossingStrategy**: Strategy for the desired direction

- **model_id**: operational model
- **cell_size**: Cell size
- **router_id**: Router

Example:

```
<numCPU>[3,4]</numCPU>
<seed>range(1, 10)</seed>
```

3. run the script makeini.py with the obligatory option **-f**: Using the aforementioned example the call is:

```
python makeini.py -f test_case1/inifile.xml
```

The script creates two directories:

- **test_case/trajectories** will be the location of the trajectories
- **test_case/inifiles** will be the location of the project files, that will be produced based on the master inifile (in this case test_case1/inifile.xml). Note, that the geometry file and the trajectory files are all relative to the project files in the directory *inifiles*.

4.2 Pre-defined tests

This section describes the general tests that should be passed by the framework in order to guarantee a minimal usability. We are continuously adding more tests.

4.2.1 Test 1: One pedestrian moving in a corridor

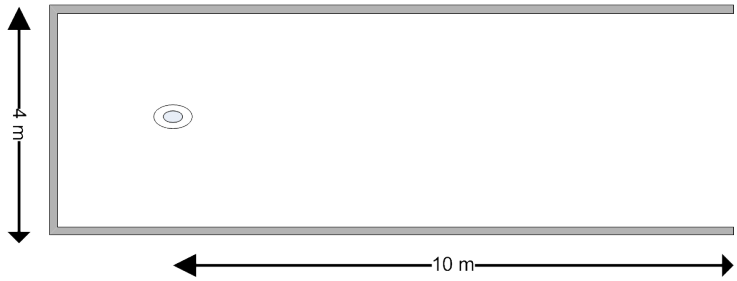


Figure 4.1: Test 1 horizontal corridor

Expected result: Pedestrian should exit the corridor within 10 seconds, if the desired speed is set to 1 m/s.

4.2.2 Test 2: One pedestrian moving in a corridor

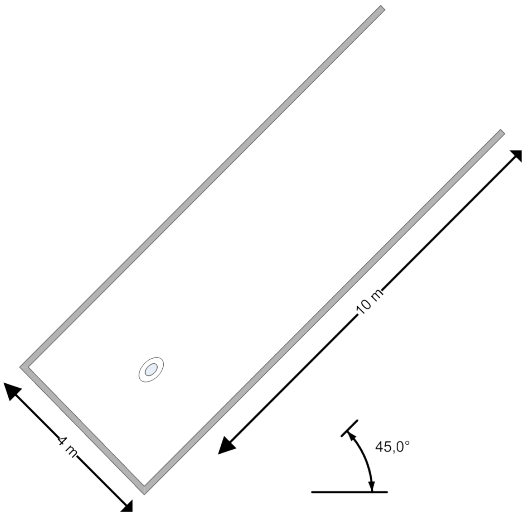


Figure 4.2: Test 2 inclined corridor 45 °(2D)

Expected result: Pedestrian should exit the corridor within 10 seconds, if the desired speed is set to 1 m/s.

4.2.3 Test 3: One pedestrian moving in a corridor with a desired direction

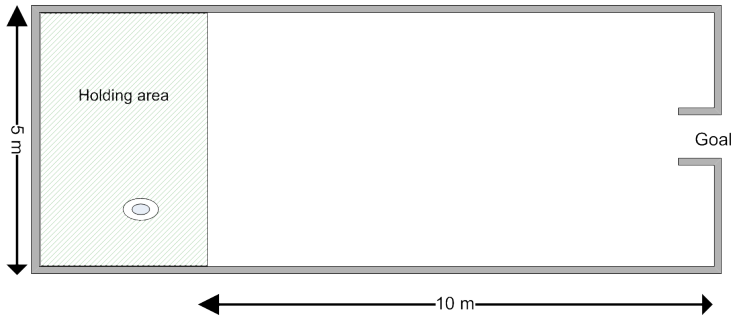


Figure 4.3: Test 3 pedestrian moving in a corridor with a desired direction

Expected result: from any random starting position in the holding area, the pedestrians should reach the marked goal.

4.2.4 Test 4: single pedestrian moving in a corridor with an obstacle

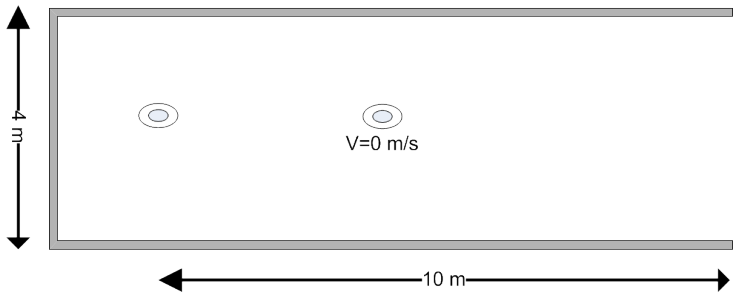


Figure 4.4: Test 4

Expected result: Pedestrian left should be able to overtake the standing pedestrian

4.2.5 Test 5: single pedestrian moving in a very narrow corridor with an obstacle

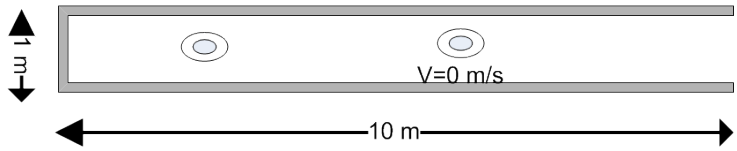


Figure 4.5: Test 5 single pedestrian moving in a very narrow corridor with an obstacle

Expected result: Pedestrian left should stop without overlapping with the standing pedestrian.

4.2.6 Test 6: single pedestrian moving in a corridor with more that one target

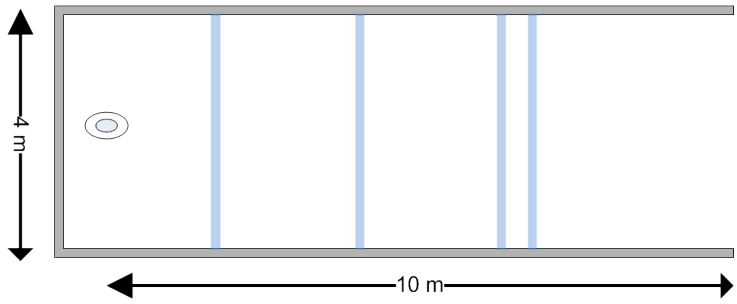


Figure 4.6: Test 6 pedestrian moving in a corridor with more that one target

Expected result: The pedestrian should move to through the different targets without a substantial change in the velocity i.e. with a desired speed of 1 m/s the distance of 10 m should be covered in 10 s.

4.2.7 Test 7: route choice with many exits

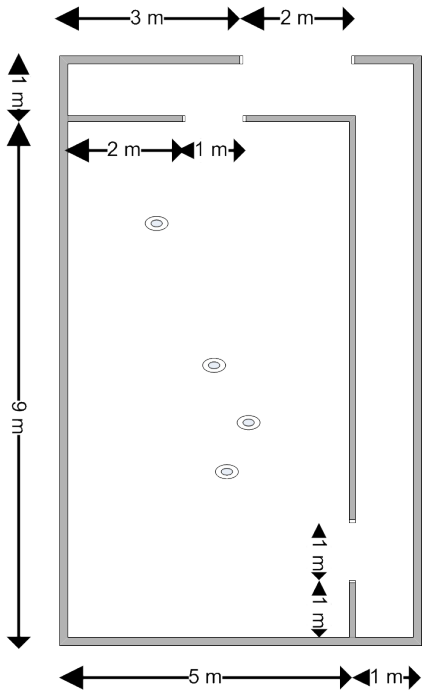


Figure 4.7: Test 7 route choice with many exits

Expected result: The pedestrians should be able to choose between the local and the global shortest paths as global route choice strategy.

4.2.8 Test 8: visibility and obstacle

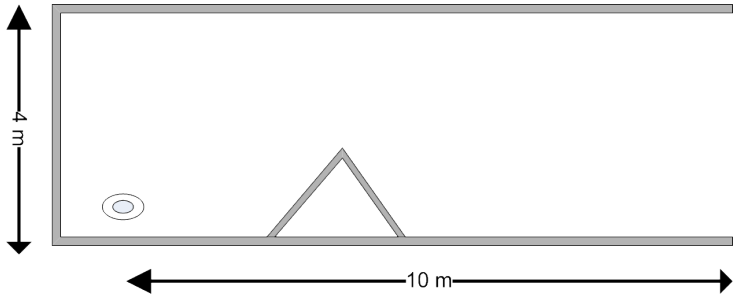


Figure 4.8: Test 8 visibility and obstacle

Expected result: The pedestrian should avoid the obstacle.

4.2.9 Test 9: runtime optimization using OpenMP

Expected results:

1. The simulation results (flow,...) should be the same for the same initial conditions independent on the number of cores used.
2. The run time should scale with the number of cores.

4.2.10 Test 10: runtime optimization using neighborhood list

Expected results:

1. The simulation results (flow, ...) should be the same, for the same initial conditions, with cell size bigger than the cut-off radius of the force-based model used.
2. The simulation should fail for a cell size smaller than the cut-off radius of the forces.
3. The simulation time should scale with the cell size.

4.3 Validation with empirical data

This section describes the tests cases that are compared with empirical data. It is still under development.

Chapter 5

JPSvis

JPSvis is the tool used for visualizing the trajectories. It is available on Linux, Windows and OS X platforms. It features a simple and streamlined user interface build with Qt and VTK.

5.1 Pre-compiled binaries

Pre-compiles binaries are available from our official page. After downloading and unpacking the archived, you can drag and drop a trajectory file or just the geometry file on the executable.

5.2 Compiling from sources

The packages needed for compiling are:

1. Qt Toolkit, version 4.5.3 or newer available at http://qt-project.org/.
2. VTK: Visualization Toolkit, version 5.4 or newer available from http://www.vtk.org/VTK/resources/software.html.
3. FFMPEG: Needed for video support under Linux.

Download and unpack the sources from https://github.com/JuPedSim/JuPedSim. You may need to adjust the path to vtk in the JPSvis.pro file. Then run

```
> qmake JPSvis.pro
> make
```

Bibliography

[1] Mohcine Chraibi, Ulrich Kemloh, Armin Seyfried, and Andreas Schadschneider. Force-based models of pedestrian dynamics. Networks and Heterogeneous Media, 6(3):425–442, 2011.

[2] Mohcine Chraibi, Armin Seyfried, and Andreas Schadschneider. Generalized centrifugal force model for pedestrian dynamics. Physical Review E, 82:046111, 2010.

[3] Simo Hostikka, Timo Korhonen, Tuomas Paloposki, Tuomo Rinne, Katri Matikainen, and Simo Heliövaara. Development and validation of fds+evac for evacuation simulations. Technical Report 2421, VTT Technical Research Centre of Finland, 2008.

[4] Armel Ulrich Kemloh Wagoum, Armin Seyfried, and Stefan Holl. Modelling dynamic route choice of pedestrians to assess the criticality of building evacuation. Advances in Complex Systems, 15(3), 2012.

[5] Armel Ulrich Kemloh Wagoum, Bernhard Steffen, Armin Seyfried, and Mohcine Chraibi. Parallel real time computation of large scale pedestrian evacuations. Advances in Engineering Software, 60-61:98–103, 2013. CIVIL-COMP: Parallel, Distributed, Grid and Cloud Computing.

[6] Andrea Portz and Armin Seyfried. Analyzing stop-and-go waves by experiment and modeling. In R.D. Peacock, E.D. Kuligowski, and J.D. Averill, editors, Pedestrian and Evacuation Dynamics 2010, pages 577–586. Springer, 2011.

Appendices

Appendix A

The Gompertz Model

A.1 Model definition

The Gompertz model is based on a continuous "physical" force and shares many parameters with the Generalized Centrifugal Force Model. Both models in fact only differ in the definition of the repulsive force (see Equation A.1 and Equation A.3).

Given the Gompertz function:

$$\mathcal{G}(\beta_{ij}) = \exp\big(-b\exp(-c\beta_{ij})\big), \tag{A.1}$$

with

- b the displacement along the x-axis $\equiv$ cut off radius
- c the growth rate (y scaling).

The function (A.1) depends on the quantity:

$$\beta_{ij} = 1 - \frac{d_{ij}}{r_i + r_j}, \tag{A.2}$$

whith d_{ij} the distance between pedestrians i with radius r_i and j with radius r_j.

The repulsive force is then defined as:

$$\overrightarrow{F_{ij}}^{\text{rep}} = -\eta' \cdot \parallel \overrightarrow{v_i^0} \parallel \cdot \mathcal{G}(\beta_{ij}) \cdot \overrightarrow{e_{ij}}, \tag{A.3}$$

A.2 Cutoff radius

While the cutoff radius for the GCFM is an input parameter, in the Gompertz model we have to derive it from other parameters.

For this porpoise we choose a cutoff radius β_c such that, the distance (between centers) is equal to a multiple of the sum of radii,

$$\mathcal{G}(\beta < \beta_c) \ll 0.$$

Considering Eq. (A.2) we set:

$$\beta_c = -2.$$

(corresponding to $3\times$ the sum of radii)

Remark:

$\beta < \beta_c$ and $d - r_1 - r_2 = -(r_1 + r_2)\beta$ imply

$$d - r_1 - r_2 \geq \underbrace{-2\mathrm{max}(\mathrm{E_a}, \mathrm{E_b})\beta_\mathrm{c}}_{\text{cutoff}}. \tag{A.4}$$

E_a and E_b are resp. the semi-axes in the direction of motion and orthogonal to it.

We have

$$\max(E_a) = a_{\min} + a_\tau v^0$$

and

$$\max(E_b) = 0.5(b_{\min} + 0.49)$$

For $a_{\min} = 0.18$ m, $a_\tau = 0.23$ s and $v^0 = 1.34$ m/s we get:

$$\max(E_a) = 0.488\ m$$

And for $b_{\min} = 0.4$ m we get:

$$\max(E_b) = 0.445\ m$$

These values imply for Eq. (A.4) that the cutoff radius for the effective is about 2 m (0.488· 2· 2 = 1.952).

A.3 Model Calibration

We want to choose b and c such that the following inequalities are fulfilled

$$\mathcal{G}(\beta < \beta_c) < \epsilon \tag{A.5}$$

and

$$\mathcal{G}(\beta > \beta_m) > \Theta, \tag{A.6}$$

β_c being the cutoff radius and β_m the maximum allowed β_{ij} reducing the amount of possible overlapping between pedestrians i and j. Θ is a threshold and ϵ is a small constant.

From Eqs. (A.5) and (A.6) we have:

$$\begin{cases} \exp\big(-b\exp(-c\beta_c)\big) = \epsilon \rightarrow -b\exp(-c\beta_c) = \log(\epsilon) & : (E) \\ \exp\big(-b\exp(-c\beta_m)\big) = \Theta \rightarrow -b\exp(-c\beta_m) = \log(\Theta) & : (F) \end{cases}$$

(E)/(F):

$$\exp(-c\beta_c) \cdot \exp(c\beta_m) = K,$$

with $K = \log(\epsilon)/\log(\Theta)$.

$$\exp(-c(\beta_c - \beta_m)) = K,$$

$$c = \frac{\log(K)}{\beta_m - \beta_c}.$$

(E)+(F):

$$-b\big(\exp(-c\beta_c) + \exp(-c\beta_m)\big) = \log(\epsilon) + \log(\Theta).$$

$$b = -\frac{\log(\epsilon \cdot \Theta)}{\exp(-c\beta_c) + \exp(-c\beta_m)}.$$

Simple calculation yields:

```
>> b=0.034229, c=2.450932
```

See Figure A.1 for the shape of the repulsive force using the aforementioned calculation of b and c.

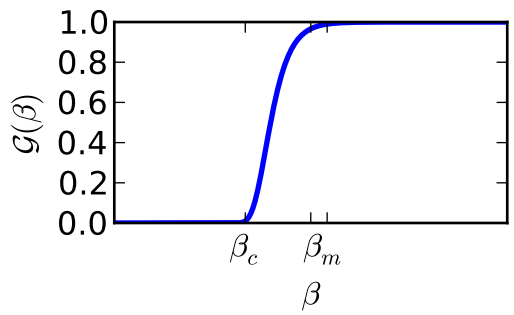
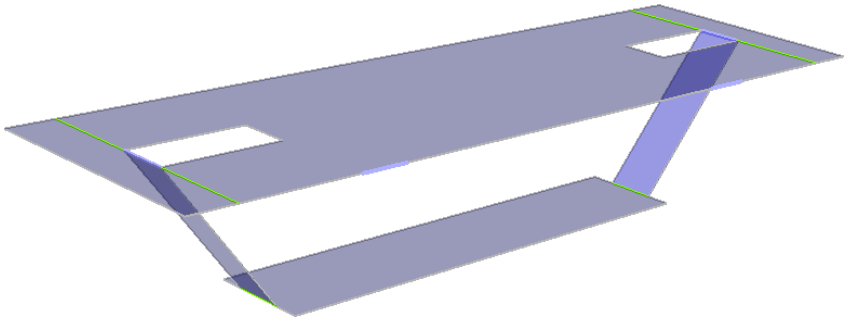


Figure A.1: The repulsive force with respect to β for b=0.034229, c=2.450932.

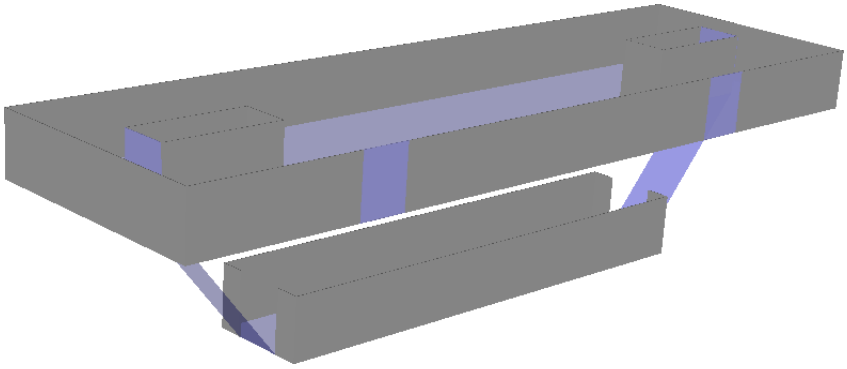
Appendix B

Showcases

B.1 Stairs geometry



(a) 3D view of the floor



(b) 3D view displaying walls

Figure B.1: Sample geometry with stairs.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<geometry version="0.5" caption="Room with two stairs">
    <rooms>
        <room id="0" caption="subway">
            <subroom id="0" class="subroom" A_x="0" B_y="0" C_z="0">
                <polygon caption="wall">
                    <vertex px="0.0" py="1.0" />
                    <vertex px="0.0" py="0.0" />
                    <vertex px="20.0" py="0.0" />
                    <vertex px="20.0" py="1.0" />
                </polygon>
                <polygon caption="wall">
                    <vertex px="20.0" py="3.0" />
                    <vertex px="20.0" py="4.0" />
                    <vertex px="0.0" py="4.0" />
                    <vertex px="0.0" py="3.0" />
                </polygon>
            </subroom>
            <subroom id="1" class="stair" A_x="-1.2" B_y="0" C_z="0">
                <polygon caption="wall">
                    <vertex px="0.0" py="1.0" />
                    <vertex px="-5.0" py="1.0" />
                </polygon>
                <polygon caption="wall">
                    <vertex px="0.0" py="3.0" />
                    <vertex px="-5.0" py="3.0" />
                </polygon>
                <up px="-5.0" py="2" />
                <down px="0.0" py="2" />
            </subroom>
            <subroom id="2" class="stair" A_x="1.2" B_y="0" C_z="-24">
                <polygon caption="wall">
                    <vertex px="20.0" py="1.0" />
                    <vertex px="25.0" py="1.0" />
                </polygon>
                <polygon caption="wall">
                    <vertex px="20.0" py="3.0" />
                    <vertex px="25.0" py="3.0" />
                </polygon>
                <up px="25.0" py="2" />
                <down px="20.0" py="2" />
            </subroom>
            <crossings>
                <crossing id="0" subroom1_id="0" subroom2_id="1">
                    <vertex px="0.0" py="1.0" />
                    <vertex px="0.0" py="3.0" />
                </crossing>
                <crossing id="1" subroom1_id="0" subroom2_id="2">
                    <vertex px="20.0" py="1.0" />
                    <vertex px="20.0 " py="3.0" />
                </crossing>
            </crossings>
        </room>
        <room id="1" caption="Hall">
            <subroom id="0" class="subroom" A_x="0" B_y="0" C_z="6">
                <polygon caption="wall">
                    <vertex px="-5.0" py="7.0" />
                    <vertex px="25.0" py="7.0" />
                </polygon>
                <polygon caption="wall">
                    <vertex px="2.0" py="-3.0" />
                    <vertex px="18.0" py="-3.0" />
```

```
                </polygon>
                <polygon caption="wall">
                    <vertex px="-5.0" py="-3.0" />
                    <vertex px="0.0" py="-3.0" />
                </polygon>
                <polygon caption="wall">
                    <vertex px="20.0" py="-3.0" />
                    <vertex px="25.0" py="-3.0" />
                </polygon>
                <polygon caption="wall">
                    <vertex px="25.0" py="1.0" />
                    <vertex px="20.0" py="1.0" />
                    <vertex px="20.0" py="3.0" />
                    <vertex px="25.0" py="3.0" />
                </polygon>
                <polygon caption="wall">
                    <vertex px="-5.0" py="1.0" />
                    <vertex px="0.0" py="1.0" />
                    <vertex px="0.0" py="3.0" />
                    <vertex px="-5.0" py="3.0" />
                </polygon>
            </subroom>
            <subroom id="1" class="subroom" A_x="0" B_y="0" C_z="6">
                <polygon caption="wall">
                    <vertex px="-5.0" py="-3.0" />
                    <vertex px="-7.0" py="-3.0" />
                    <vertex px="-7.0" py="7.0" />
                    <vertex px="-5.0" py="7.0" />
                </polygon>
            </subroom>
            <subroom id="2"  class="subroom" A_x="0" B_y="0" C_z="6">
                <polygon caption="wall">
                    <vertex px="25.0" py="-3.0" />
                    <vertex px="27.0" py="-3.0" />
                    <vertex px="27.0" py="7.0" />
                    <vertex px="25.0" py="7.0" />
                </polygon>
            </subroom>
            <crossings>
                <crossing id="2" subroom1_id="0" subroom2_id="2">
                    <vertex px="25.0" py="-3.0" />
                    <vertex px="25.0" py="1.0" />
                </crossing>
                <crossing id="3" subroom1_id="0" subroom2_id="2">
                    <vertex px="25.0" py="3.0" />
                    <vertex px="25.0 " py="7.0" />
                </crossing>
                <crossing id="4" subroom1_id="0" subroom2_id="1">
                    <vertex px="-5.0" py="-3.0" />
                    <vertex px="-5.0 " py="1.0" />
                </crossing>
                <crossing id="5" subroom1_id="0" subroom2_id="1">
                    <vertex px="-5.0" py="3.0" />
                    <vertex px="-5.0 " py="7.0" />
                </crossing>
            </crossings>
        </room>
    </rooms>

    <transitions>
        <transition id="6" caption="No_Name 1" type="emergency" room1_id="0"
              subroom1_id="1" room2_id="1" subroom2_id="1">
```

```
124             <vertex px="-5.0" py="1.0" />
125             <vertex px="-5.0" py="3.0" />
126         </transition>
127         <transition id="7" caption="No_Name 2" type="emergency" room1_id="0"
                subroom1_id="2" room2_id="1" subroom2_id="2">
128             <vertex px="25.0" py="1.0" />
129             <vertex px="25.0" py="3.0" />
130         </transition>
131         <transition id="8" caption="No_Name 3" type="emergency" room1_id="1"
                subroom1_id="0" room2_id="-1" subroom2_id="-1">
132             <vertex px="0.0" py="-3.0" />
133             <vertex px="2.0" py="-3.0" />
134         </transition>
135         <transition id="9" caption="No_Name 4" type="emergency" room1_id="1"
                subroom1_id="0" room2_id="-1" subroom2_id="-1">
136             <vertex px="18.0" py="-3.0" />
137             <vertex px="20.0" py="-3.0" />
138         </transition>
139     </transitions>
140 </geometry>
```