Phantasma Simulation Challenge: Development of a Solver for Pedestrian Simulation

Mridul Mani Tripathi
mridul.tripathi@rwth-aachen.de

Abstract

This project aims to simulate the microscopic model of pedestrian flow in intersecting corridors. The report provides the details about the approach taken for modeling the pedestrian flow and the results obtained by varying the flux values.

1 Problem description

The geometry for the corridors in shown in the figure below.

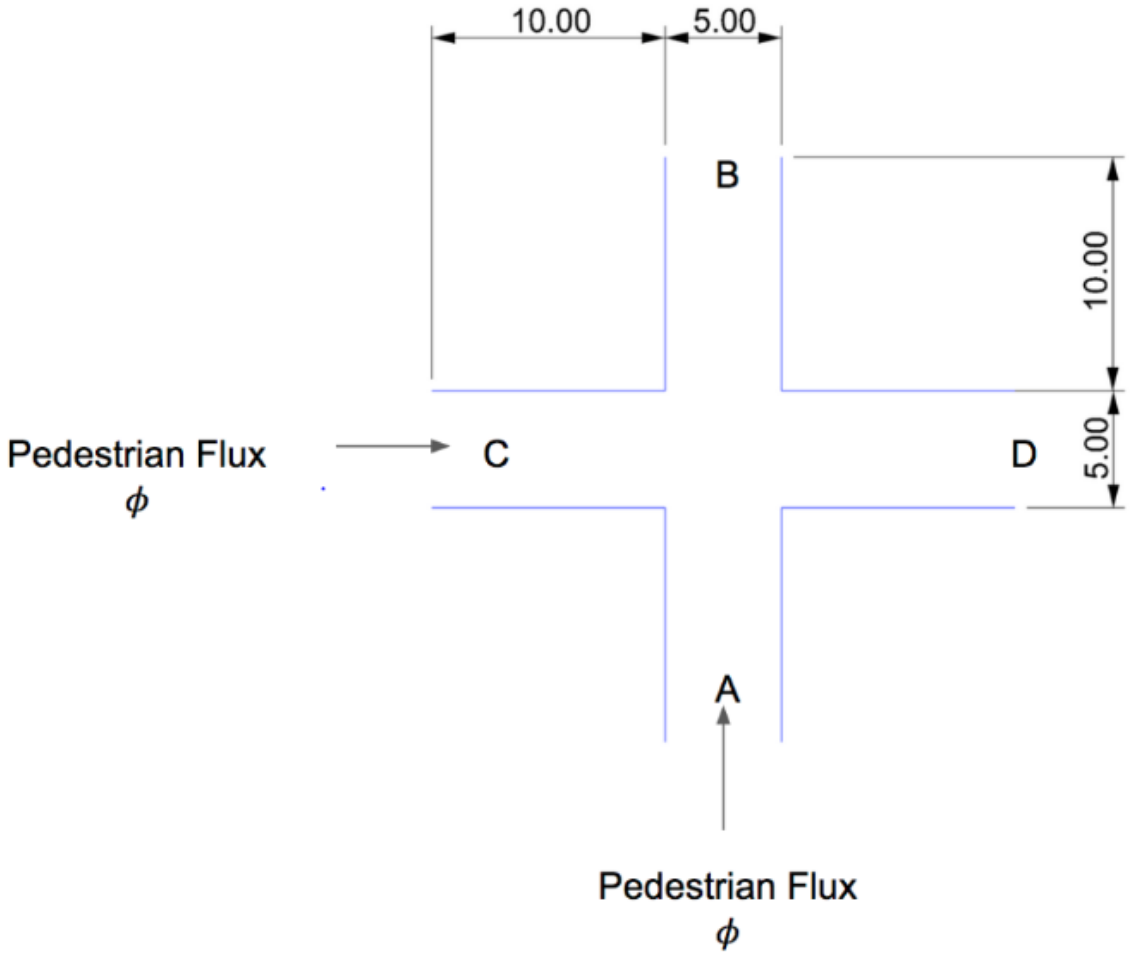

Figure 1: Geometry of corridors (all measurements are in metres) [1]

Pedestrians with a constant flux ϕ enter the corridor from the bottom (region A) and left (region C). The pedestrians entering from region A leave the corridor at region

B and the pedestrians entering from region C leave the corridor at region D. The task is to set up a solver for simulating the pedestrian flow such that individual pedestrian is represented.[1]

The pedestrians have circular cross-section with radius of 0.25 m, and their velocity is in the range of 1.1 - 1.3 m/s. The maximum acceleration that a pedestrian can attain is 2 m/s^2. The motion is integrated using a desired time step for 60 seconds and basic plots for varying pedestrian fluxes is generated. [1]

2 Models

2.1 Corridor model

The corridors are generated by taking start and end coordinates of the corridor as an input from the user. The orientation of the corridors is computed using the X and Y values of the start and end coordinates. The rectangular corridors are generated on the screen using the pygame library. The width of the corridors is set to 5 m.

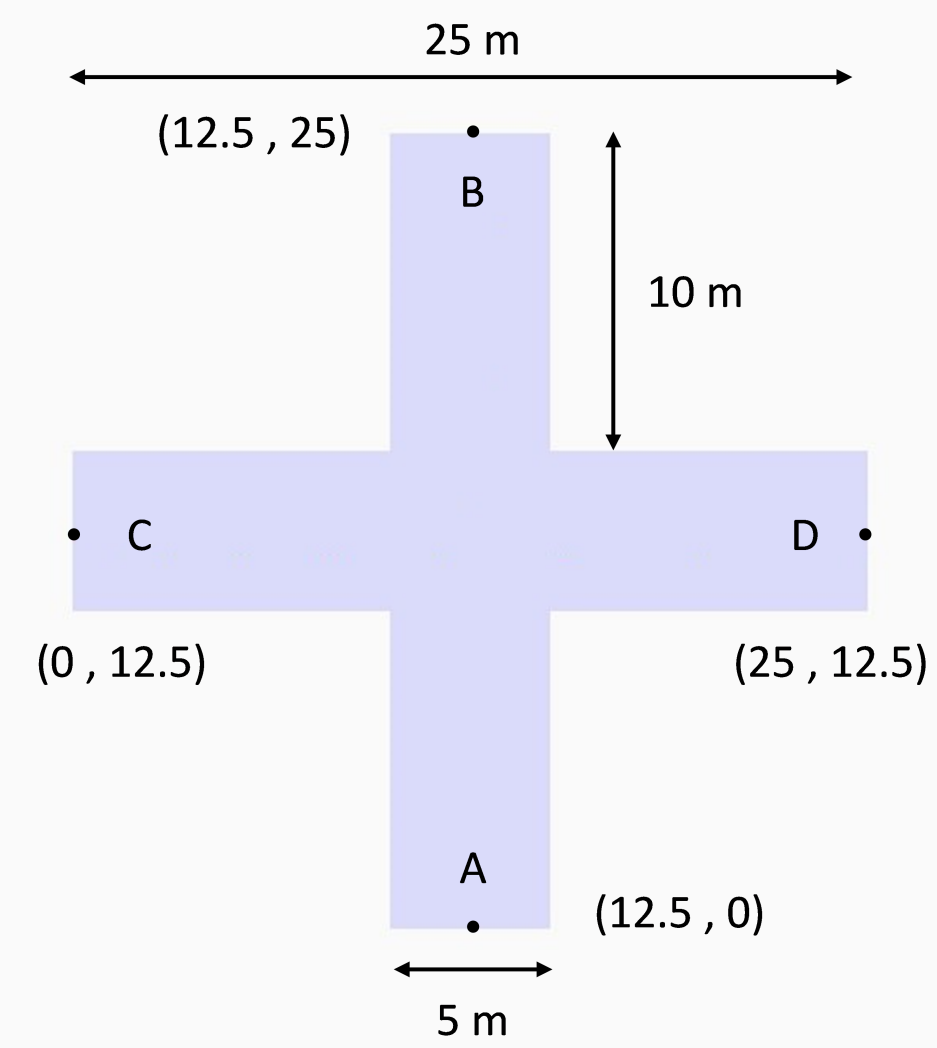

Figure 2: Geometry of corridors

The pedestrians are provided as the double ended queue to a corridor, so that they can added/removed from either ends.

2.2 Pedestrian model

The pedestrians have circular cross section. For updating the position and velocity of the pedestrian, the equations of the motion are used. The parameters for their models are listed below.

- ◇ Radius of the pedestrian = $0.25m$
- ◇ Length of the pedestrian = $0.5m$
- ◇ Minimum distance between two pedestrians = $0.25m$
- ◇ Maximum velocity of the pedestrian = $1m/s$
- ◇ Maximum acceleration of the pedestrian = $2m/s^2$
- ◇ Velocity += Acceleration * Time step
- ◇ Position += Velocity * Time step + Acceleration * Time step^2
- ◇ The pedestrians are uniformly accelerating with the acceleration of $2m/s^2$.

The rate of generation of the pedestrians is provided by the user as the input (flux). Since the pedestrians are provided as the double ended queue to the corridors, the pedestrians are generated for each corridor separately. The pedestrians are added to each corridor alternatively so that their collision can be avoided at the intersection of the corridors. The pedestrians are added if two conditions are satisfied:

- ◇ The time elapsed since the last addition of the pedestrian is more than the desired value based on the flux rate.
- ◇ The position of the latest pedestrian is greater than the minimum distance required between the two pedestrians.

The circular pedestrians are drawn on the screen using pygame library for each corridor.

3 Simulation

The simulation is performed for 60 seconds with time step of 1 second. First the corridors are created using the input from the user. For each time step, pedestrians are updated for both corridors and if a pedestrian crosses the bounds of the corresponding corridor, it is removed. Thus the corridors are also updated in each time step.

4 Visualization screen

The visualization is performed using the pygame library. The size of the screen is set to 700 x 700 with 20 times zoom so that the corridors and the pedestrian flow is clearly visible. The screen updates at the regular interval of 1 ms. The parameters from the simulation are passed to the screen for visualization. The screen shows the corridors, pedestrians, time elapsed during the simulation, and the number of pedestrians generated.

5 Results

This section provides the plots for number of pedestrians generated at each second during the simulation. The image of the visualization screen for the last time step is also shown.

5.1 Flux = 0.2 pedestrians per second for each corridor

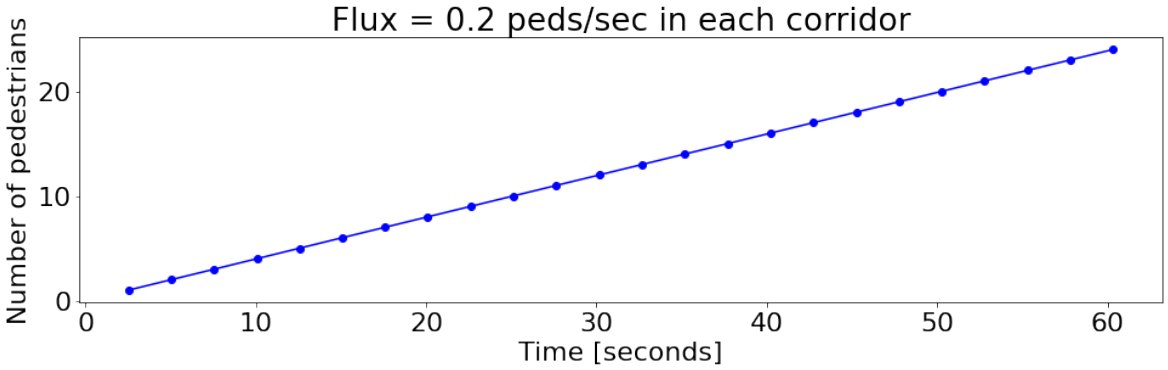

Figure 3: Total number of pedestrians generated at each second

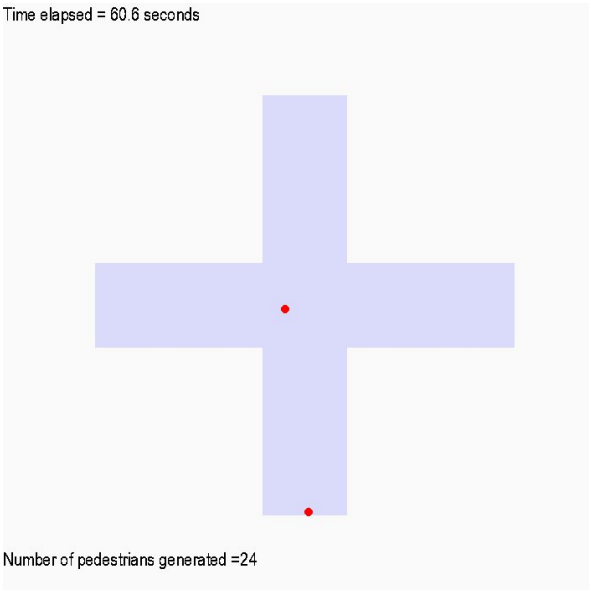

Figure 4: Snapshot of the screen for the last time step

5.2 Flux = 0.6 pedestrians per second for each corridor

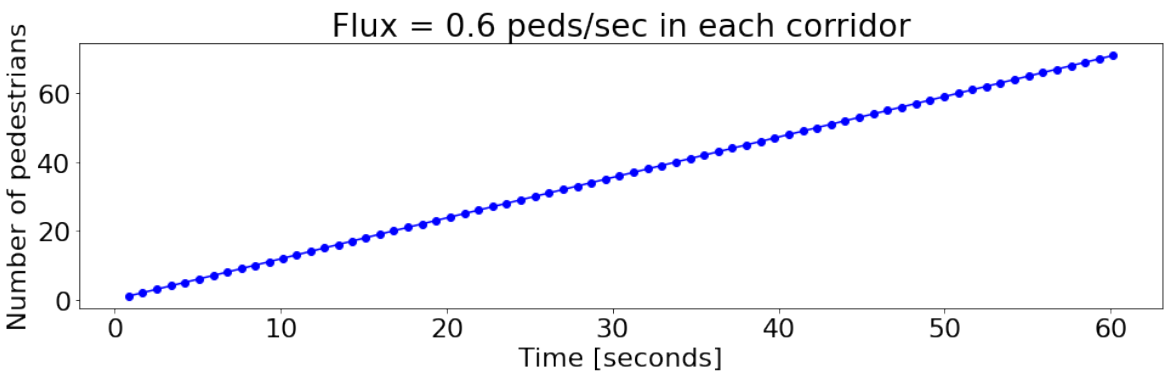

Figure 5: Total number of pedestrians generated at each second

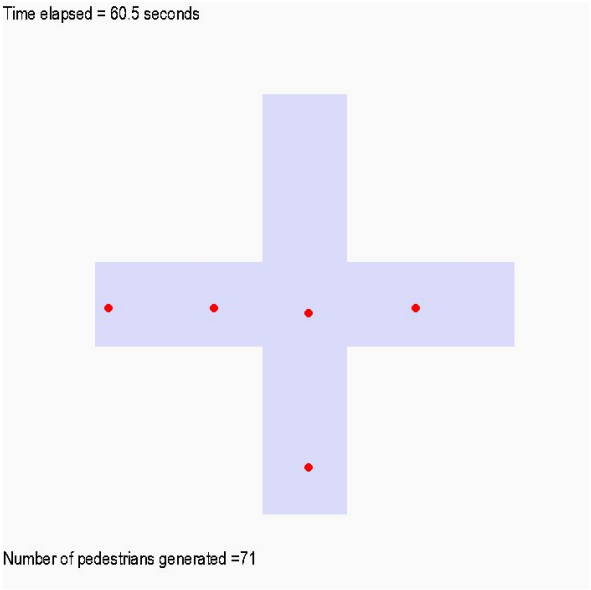

Figure 6: Snapshot of the screen for the last time step

5.3 Flux = 1.2 pedestrians per second for each corridor

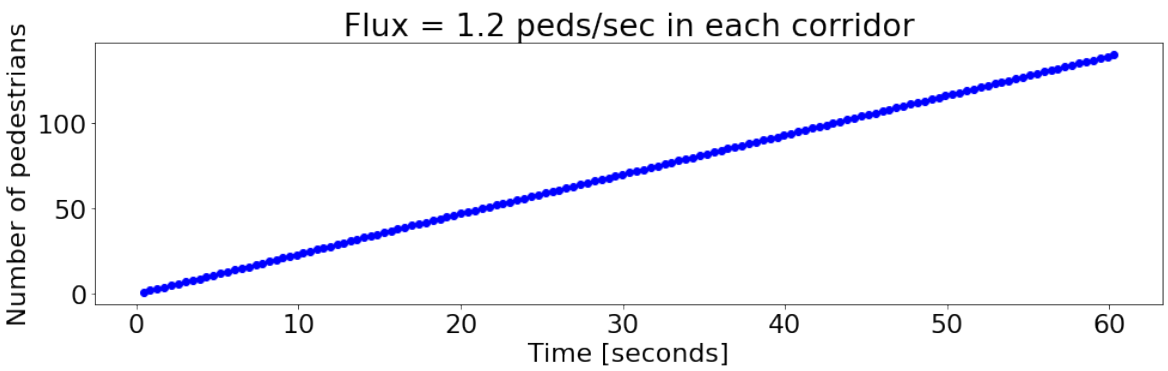

Figure 7: Total number of pedestrians generated at each second

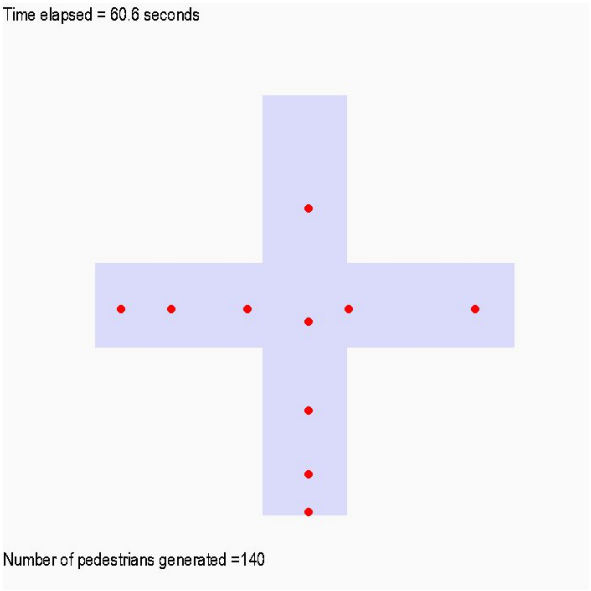

Figure 8: Snapshot of the screen for the last time step

5.4 Flux = 2 pedestrians per second for each corridor

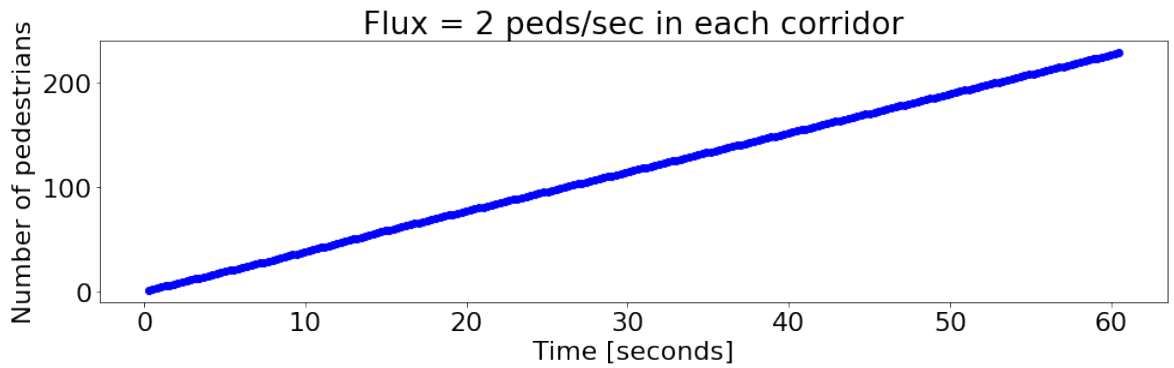

Figure 9: Total number of pedestrians generated at each second

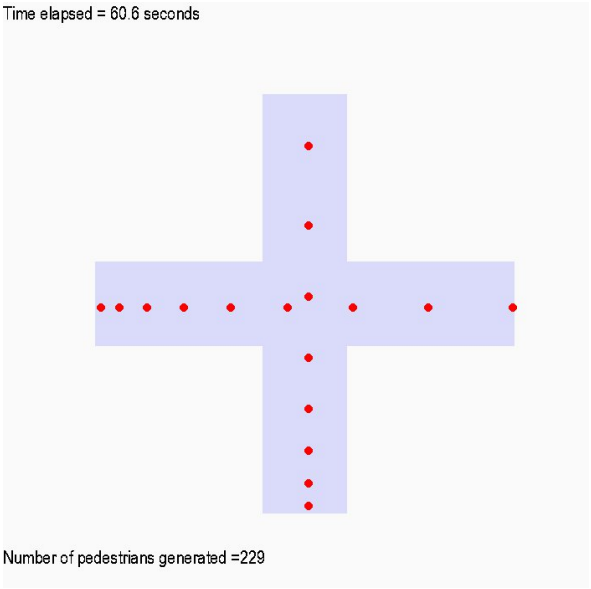

Figure 10: Snapshot of the screen for the last time step

The code for the simulation is provided on the GitHub repository.[2] The simulation is based on the tutorial provided by Bilal Himite. [3]

6 Further improvements

- ◇ The pedestrians generated for large flux rates are smaller than the required value. A different algorithm needs to be developed to accommodate the necessary changes.
- ◇ The acceleration and the velocity of the pedestrians are set to their maximum value at each update. If these parameters are kept variable, collisions occur between pedestrians.
- ◇ The basic equation of motions are used for modelling the pedestrians. A deceleration can also be added to make the model more realistic.

- ◇ To avoid collisions among pedestrians from different corridors, a different approach is needed based on the current positions of the pedestrians rather than adding pedestrians at alternatively.

References

[1] Problem description provided by Phantasma Labs.

[2] Mridul Mani Tripathi, PL_PedSim,
URL: https://github.com/mridulmanitripathi/PL_PedSim

[3] Bilal Himite, Simulating Traffic Flow in Python,
URL: https://towardsdatascience.com/simulating-traffic-flow-in-python-ee1eab4dd20f