The Journal of Open Source Software

PySocialForce: A Python Implementation of the Extended Social Force Model for Pedestrian Dynamics

Yuxiang Gao[1] **and Chien-Ming Huang**[1]

1 Department of Computer Science, The Johns Hopkins University

DOI: 10.21105/joss.02494

Software

- Review
- Repository
- Archive

Editor: Pierre de Buyl

Reviewers:

- @ixjlyons
- @xuqiancheng

Submitted: 06 July 2020
Published: 21 July 2020

License
Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License (CC BY 4.0).

Summary

Modeling pedestrian dynamics has a variety of valuable applications, ranging from emergency simulation and urban planning to crowd simulation in video games and movies. Pedestrian simulation also plays an important role in developing mobile robots that are capable of navigating crowded human environments in a safe, efficient, and socially appropriate manner; these robots promise to bring practical robotic assistance—such as emergency response, parcel delivery, and guided tours—to our daily lives. While robot navigation has been widely studied, *socially acceptable* navigation in crowded environments remains a challenge due to the rich dynamics of social interactions between pedestrians and the limited human data available for training and validating social navigation algorithms.

Crowd simulation offers an alternate source of data for developing modern machine learning algorithms for social navigation. One popular model for pedestrian dynamics is the social force model proposed by Helbing & Molnár (1995), which suggests that pedestrian behaviors can be modeled as if they are subject to "social forces." To simulate social groups, which constitute most human crowds, Moussaïd, Perozo, Garnier, Helbing, & Theraulaz (2010) extended this model by introducing additional forces for social groups.

`PySocialForce` is a pure Python package for simulating crowd dynamics based on the extended social force model. While it can be used for general crowd simulation, it is designed with social navigation applications in mind. Our Python implementation makes it easily extensible (e.g., adding new custom "social forces") and able to interface with modern reinforcement learning environments (e.g., OpenAI Gym).

The `PySocialForce` Package

`PySocialForce` implements the social force model described in (Moussaïd et al., 2009) and its extension with social groups described in (Moussaïd et al., 2010). The package started as a fork of `socialforce` (Kreiss, 2018), which implements the original social force model proposed in (Helbing & Molnár, 1995); however, we ended up rewriting the package to use the extended version of the social force model as described in (Moussaïd et al., 2009, 2010) with its core functions accelerated with just-in-time compilation using Numba. We also drew inspiration from the `pedsim_ros` (Okal & Linder, 2014) package, an ROS package that implements the extended social force model. However, as it is a GUI application and has dependency on both ROS and QT, we have found it difficult to run `pedsim_ros` on servers without the X Window System.

In our implementation, we calculate six default forces. Three of them (Moussaïd et al., 2009) are for individuals:

1. the repulsive forces between pedestrians;
2. the attractive forces between each pedestrian and their goal(s);
3. the repulsive forces from obstacles.

The other three forces (Moussaïd et al., 2010) are for groups:

4. the coherence force that holds group members together;
5. the repulsive force that keeps members from getting too close to each other;
6. a force calculated from the gaze directions of pedestrians to maintain group formations.

Users can easily create their own forces by inheriting the `Force` metaclass.

To use `PySocialForce`, the user passes in the initial states—the positions, velocities, and goals of the pedestrians—and the optional information of social groups and obstacles. Input parameters can be passed in as a `toml` file. Given the necessary initial state information, a typical example of running the simulator for 50 timesteps is shown in the example below:

```
import pysocialforce as psf
simulator = psf.Simulator(initial_state,
                          groups=groups,
                          obstacles=obstacles,
                          config_file="my_config.toml")
simulator.step(50)
```

After the simulation finishes, one can easily generate an animation of the simulation with the provided `SceneVisualizer` context:

```
with psf.plot.SceneVisualizer(simulator, "output_image") as sv:
    sv.animate()
```

The result of this example is shown in Figure 1. In this example, a two-person group encounters a single pedestrian coming from the other direction and adjusts accordingly, all while avoiding an obstacle in their path. The group members are interconnected with a purple line. Figure 2 shows another example with a corridor setup.

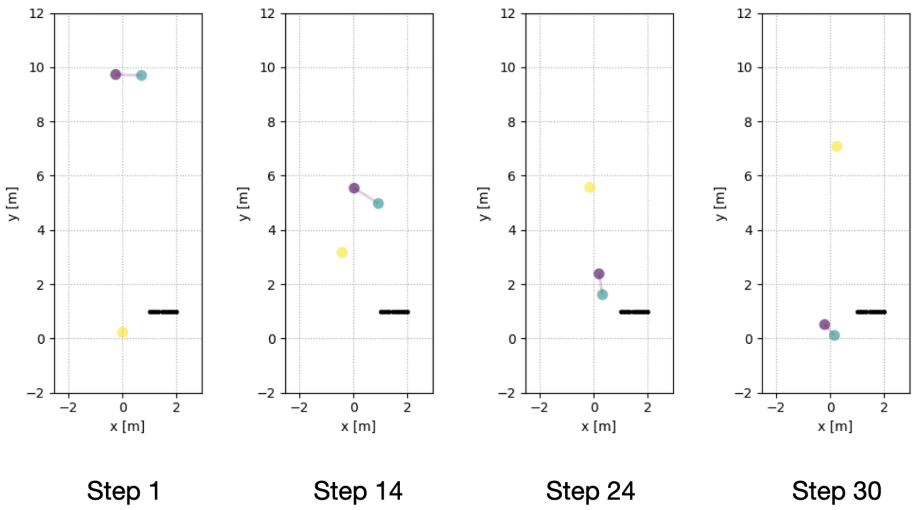

Figure 1: Example simulation.

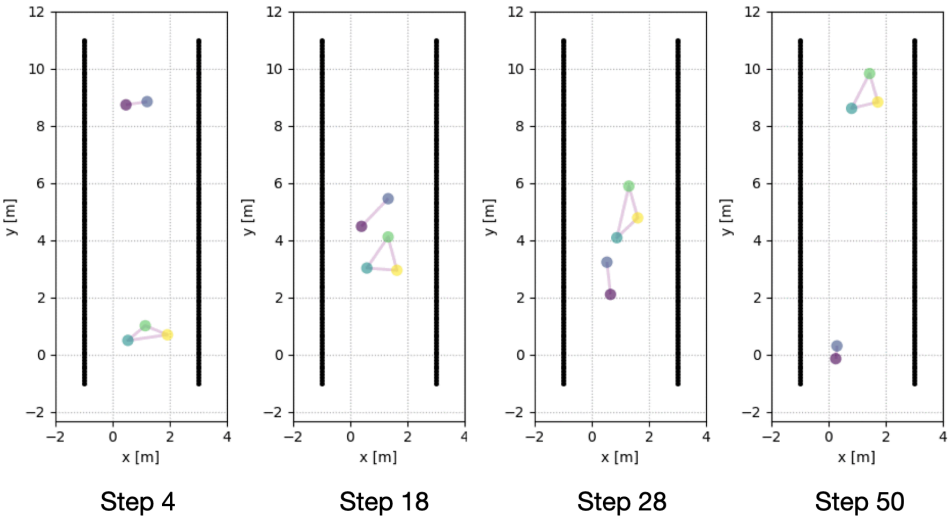

Figure 2: Two groups passing each other in a narrow corridor.

Acknowledgments

This project is supported by the Johns Hopkins University Institute for Assured Autonomy.

References

Helbing, D., & Molnár, P. (1995). Social force model for pedestrian dynamics. *Physical Review E*, *51*(5), 4282–4286. doi:10.1103/PhysRevE.51.4282

Kreiss, S. (2018). Socialforce. *GitHub repository*. GitHub. Retrieved from https://github.com/svenkreiss/socialforce

Moussaïd, M., Helbing, D., Garnier, S., Johansson, A., Combe, M., & Theraulaz, G. (2009). Experimental study of the behavioural mechanisms underlying self-organization in human crowds. *Proceedings of the Royal Society B: Biological Sciences*, *276*(1668), 2755–2762. doi:10.1098/rspb.2009.0405

Moussaïd, M., Perozo, N., Garnier, S., Helbing, D., & Theraulaz, G. (2010). The walking behaviour of pedestrian social groups and its impact on crowd dynamics. *PLoS ONE*, *5*(4), 1–7. doi:10.1371/journal.pone.0010047

Okal, B., & Linder, T. (2014). Pedsim_ros. *GitHub repository*. GitHub. Retrieved from https://github.com/srl-freiburg/pedsim_ros