# JoSS
The Journal of Open Source Software

PySwarming: a research toolkit for Swarm Robotics

Emerson Martins de Andrade [1,2], **Antonio Carlos Fernandes** [1,2], **and Joel Sena Sales Junior** [1,2]

1 Federal University of Rio de Janeiro, Rio de Janeiro, Brazil **2** Ocean Engineering Program, Laboratory of Waves and Current, LOC/COPPE/UFRJ, Rio de Janeiro, Brazil

Summary

When considering a system composed of a group of robots, swarm robotics is an approach that can be used to coordinate this group. These swarms can be inspired or not by social insects or other animal societies (Trianni, 2008), where basic behaviors are usually used to compose complex tasks. These previously mentioned basic behaviors have been studied for a long time, with applications, for example, to flocks, herds, and schools (Reynolds, 1987) and multi-robot teams (T. Balch & Arkin, 1998). Here we introduce PySwarming, a tool that makes easy the coordination of swarms and serves as a centerpiece, organizing different methods developed in the swarm robotics field. Its flexibility (written in Python) and customizability (easily customized by users) encourage interaction and scientific progress in the research community.

Introduction

Controlling a system composed of a group of robots can be a challenge, then, swarm robotics is an approach that is widely used to coordinate this kind of system. Furthermore, the changes that the field of swarm robotics has experienced in the last decade are unprecedented, with various demonstrations showing the potential of this technology (Dorigo et al., 2021). Moreover, it is important to note that swarm robotics is a subfield of multi-robot systems, itself a subfield of mobile robot research (Dias et al., 2021). Additionally, these swarms may or may not be inspired by social insects and other animal societies (Trianni, 2008), where basic behaviors are typically used to compose complex tasks. The aforementioned behaviors have long been studied and applied, for instance, to herds, flocks, schools, etc. (Reynolds, 1987; Toner & Tu, 1998), self-driven particles (Vicsek et al., 1995), large collections of robots (Reif & Wang, 1999; Spears & Gordon, 1999), and multi-robot teams (T. Balch & Arkin, 1998).

Related Software Packages

For more than 20 years software for swarm robotics has been created, adapted, and tested in different ways (Calderón-Arce et al., 2022). Software packages that allow the creation of virtual scenarios and swarm robots with sensing, processing, and actuating capabilities are crucial for the swarm robotics field. These software packages may be split into two categories: (1) *Swarm Simulators*, which in general can simulate swarm robots with sensors and actuators, and (2) *Behavior Packages*, which are composed of a bunch of ready-to-use collective behaviors. Also, these software are written in different programming languages, which may be an initial barrier for new users, depending on how much this language is difficult to learn or previous knowledge by the user.

Concerning swarm robotics, (Calderón-Arce et al., 2022) presents a good review regarding swarm robotics simulators, platforms, and applications. Therefore, for software packages we have, for instance: (1) Buzz, which is a programming language for heterogeneous robot swarms

(C. Pinciroli et al., 2015). It offers primitives to define swarm behaviors and also single-robot instructions; (2) ChoiRbot by (Testa et al., 2021), which is a toolbox for distributed cooperative robotics based on the Robot Operating System (ROS) 2; (3) ROS2Swarm, which is a package for applications of swarm robotics that provides a library of ready-to-use swarm behavioral primitives (Kaiser et al., 2022); (4) ARGoS, which is a multi-physics robot simulator, able to simulating large-scale swarms of robots (Carlo Pinciroli et al., 2011); (5) Stage Simulator, which provides a virtual world populated by mobile robots and sensors, along with various objects for the robots to sense and manipulate (Vaughan, 2008); (6) USARSim is a free 3D simulator similar to Gazebo (Carpin et al., 2007); (7) The Swarm-bots project (Mondada et al., 2004) is also a robotic simulator with swarm-intelligence-based control mechanisms, but it is not publicly available; (8) and TeamBots is a Java-based collection of application programs and Java packages for multiagent mobile robotics research (T. R. Balch, 1998). **Table 1** shows a summary of these software and their respective category and programming languages.

Software	Category	Programing Language
Buzz	Behaviors package	Buzz
ChoiRbot	Behaviors package[1]	Python
ROS2Swarm	Behaviors package[2]	Python
ARGoS	Simulator	C++
Stage	Simulator	C++
USARSim	Simulator	UnrealScript
Gazebo	Simulator	C++ and Python[3]
Swarm-bots	Simulator and Behaviors package	Not found
TeamBots	Simulator and Behaviors package	Java and C

Table 1: Summary of software packages with their respective categories and programming languages.

Statement of Need

Concerning the *Swarm Simulators*, there is an enormous variety of excellent candidates, but it is hard to compare each other since each one has been developed with different objectives (Erez et al., 2015). Then, recommendations regarding this choice can be found in (Calderón-Arce et al., 2022). Regarding *Behavior Packages*, their objective is to offer ready-to-use collective behaviors, depending on the programming language in which they are implemented they can be used inside different simulators. However, as can noticed from the previous section, in general, different packages are written in different languages, imposing a possible barrier for new users and the use of the package inside simulators written in a different programming language. Moreover, important characteristics regarding the software are the possibility of leveraging existing datasets, develop new algorithms, and quick prototyping, which are not commonly found together.

PySwarming

Because of the lack of a cross-platform *Behavior Package*, we introduce PySwarming, which is a tool that facilitates the coordination of swarms and serves as a centerpiece, organizing different methods developed in the swarm robotics field. The package is written in Python, which is one of the most accessible languages for learning and prototyping nowadays. Also, differently from other Python-based packages such as ChoiRbot and ROS2Swarm that are designed to work with ROS, the PySwarming package is implemented in a way that makes possible

---

[1]These packages were built to be used with ROS.
[2]These packages were built to be used with ROS.
[3]These are the programming languages that can be used in the interface ROS-Gazebo.

cross-platform use. In addition, the PySwarming package comes with a simple ready-to-use simulation feature, which helps lower the barrier for newcomers to the field.

Then, considering the challenge of organizing the various methods developed in the swarm robotics field, PySwarming comes as a focal point, being flexible (written in Python) and customizable (can be easily adapted by the user), increasing the interaction of the researcher community and the advance of science. Also, PySwarming's characteristic is to make the implementations easy to read, keeping the syntax simple and closer to their sources. For example, the target algorithm by (Zoss et al., 2017) is easily comparable with the mathematical formula of their article.

PySwarming differs from Buzz and TeamBots mainly by the fact that it focuses on swarm behaviors and it is written in Python, which has a thriving ecosystem of third-party libraries. Also, implementations like ChoiRbot and ROS2SWARM require ROS to run, which makes PySwarming more suitable for obtaining swarm behaviors through different platforms other than ROS. Lastly, unlike ARGoS, Stage, and USARSim the main goal of PySwaming is not to be a simulator itself, but a common place for different swarm coordination methods and other tools.

Concerning the algorithms, PySwarming contains implementations from different authors, for instance, Leaderless Coordination (Vicsek et al., 1995), Preferred Direction (Couzin et al., 2005), Aggregation (Zoss et al., 2017), and so on. Also, these algorithms are based on different design methods (Brambilla et al., 2013), for instance, a behavior-based design like the attraction-repulsion algorithm (for details see (Spears et al., 2004)) can drive the robotic swarming employing virtual forces, where the achieved configuration relies on minimizing the system's potential energy. More explanations regarding the algorithms and their use can be found in the API and PySwarming documentation. Also, an example usage (aggregation + heading consensus + repulsion) is described below.

Simulation Example

To start our example, we will define a set of four robots assuming we have access to their positions and orientations. Initially, they are positioned far from each other, and also they have different orientations, as can be observed in Figure 1.
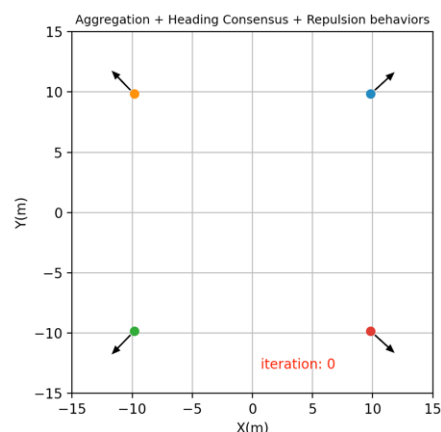
Figure 1: The initial state of the four robots. Each colored circle is a robot and the arrows indicate their orientation. The iteration number is the red text.

Then, using `PySwarming` we iterate over time by summing three different behaviors: (1) Aggregation, (2) Heading Consensus, and (3) Repulsion. Each of these behaviors is applied to each robot at each timestep. As expected, the robots will aggregate, adjust their headings,

and repulse each other simultaneously over the simulation. The intermediate and final results are shown in Figure 2.
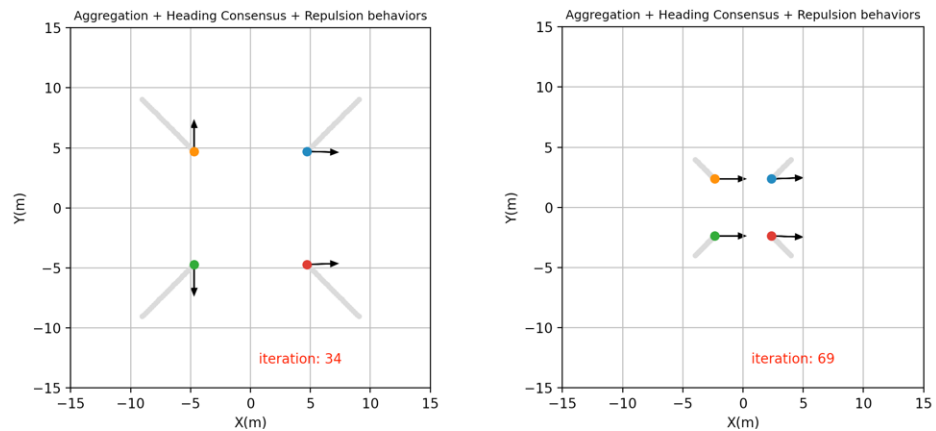
Figure 2: Intermediate (left) and final (right) state of the four robots. The gray path is the plot of the last 30 iterations of each robot.

The above simulation can be done by using other `PySwarming` behaviors, such as Attraction and Alignment, with just a few lines of code, which demonstrates the simplicity of `PySwarming`. Finally, the code for this simulation can be found in our examples directory.

How to create a new behavior and extend the package

To show PySwarming's flexibility and customizability here we illustrate a practical example. Then, imagine that we want to create a flocking behavior. Based on the literature it is known that the classical "flocking"model is composed of three terms: (1) aggregation, (2) avoidance, and (3) alignment (Zoss et al., 2017). As we have all these previous behaviors implemented in PySwarming, you can simply do:

```python
# importing the swarming behaviors
import pyswarming.behaviors as pb
```

```python
# creating a new flocking function
def flocking(*args):
    return pb.aggregation(*args) + pb.repulsion(*args) + pb.alignment(*args)
```

Then, we have created a new behavior based on existing ones. Where, *args are the arguments that will be used by these functions, for details, please see the PySwarming documentation. However, you have the freedom to extend the package and implement other models, add new features, and so on, detailed instructions can be found here.

Algorithms covered

This library includes the following algorithms to be used in swarm robotics:

- **Leaderless heading consensus**: the collective performs heading consensus (Vicsek et al., 1995);
- **Inverse power**: adjustable attraction and repulsion laws (Reif & Wang, 1999);
- **Spring**: allows the robots to maintain a desired distance between them (Reif & Wang, 1999);
- **Force law**: mimics the gravitational force (Spears & Gordon, 1999);
- **Repulsive force**: makes the individuals repulse each other (Helbing & Vicsek, 2000);

- **Body force**: introduces a body force that considers the radii of the robots (Helbing & Vicsek, 2000);
- **Inter robot spacing**: allows the robots to maintain a desired distance between them (Leonard & Fiorelli, 2001);
- **Dissipative**: a dissipative force that reduces the "energy" of the robots (Leonard & Fiorelli, 2001);
- **Leader Following**: the collective performs heading consensus with a leader (Jadbabaie et al., 2003);
- **Collision Avoidance**: the robot stays away from neighbors in the vicinity (Couzin et al., 2005);
- **Attraction and Alignment**: the robot becomes attracted and aligned (Couzin et al., 2005);
- **Preferred Direction**: the robot has a preference to move toward a preset direction (Couzin et al., 2005);
- **Lennard-Jones**: allows the formation of lattices (Carlo Pinciroli et al., 2008);
- **Virtual viscosity**: a viscous force that reduces the "oscillation" of the robots (Carlo Pinciroli et al., 2008);
- **Modified Attraction and Alignment**: the robot becomes attracted and aligned by considering a "social importance" factor (Freeman & Biro, 2009);
- **Heading Consensus**: the collective performs heading consensus (Chamanbaz et al., 2017);
- **Perimeter Defense**: the robots maximize the perimeter covered in an unknown environment (Chamanbaz et al., 2017);
- **Environment exploration**: provides spatial coverage (Chamanbaz et al., 2017);
- **Aggregation**: makes all the individuals aggregate collectively (Zoss et al., 2017);
- **Alignment**: the collective performs heading consensus (Zoss et al., 2017);
- **Geofencing**: attract the robots towards area A (Zoss et al., 2017);
- **Repulsion**: makes all the individuals repulse collectively (Zoss et al., 2017);
- **Target**: the robot goes to a specific target location (Zoss et al., 2017);
- **Area coverage**: using the Geofencing and Repulsion algorithms (Zoss et al., 2017);
- **Collective navigation**: using the Target and Repulsion algorithms (Zoss et al., 2017);
- **Flocking**: using the Aggregation, Repulsion and Alignment algorithms (Zoss et al., 2017);

Conclusion

Presented in this work is PySwarming, a package that facilitates the coordination of swarms and serves as a centerpiece, organizing different methods developed in the swarm robotics field. This package provides ready-to-use collective behaviors implementations based on the work of different authors. As shown through the examples, the fact that it is written in Python makes it more flexible and easily customizable, allowing quick prototyping. Therefore, PySwarming fills an important space, regarding the organization of the different swarming robotics methods developed over the last decade, and all the characteristics of the package confer a lower barrier for newcomers to the field, which helps to increase the interaction of the researcher community and the advancement of science.

Acknowledgements

References

Balch, T. R. (1998). *Behavioral diversity in learning robot teams*. Georgia Institute of Technology.

Balch, T., & Arkin, R. C. (1998). Behavior-based formation control for multirobot teams. *IEEE Transactions on Robotics and Automation*, *14*(6), 926–939. https://doi.org/10.1109/70.736776

Brambilla, M., Ferrante, E., Birattari, M., & Dorigo, M. (2013). Swarm robotics: A review from the swarm engineering perspective. *Swarm Intell*. https://doi.org/10.1007/s11721-012-0075-2

Calderón-Arce, C., Brenes-Torres, J. C., & Solis-Ortega, R. (2022). Swarm robotics: Simulators, platforms and applications review. *Computation*, *10*(6), 80. https://doi.org/10.3390/computation10060080

Carpin, S., Lewis, M., Wang, J., Balakirsky, S., & Scrapper, C. (2007). USARSim: A robot simulator for research and education. *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 1400–1405. https://doi.org/10.1109/ROBOT.2007.363180

Chamanbaz, M., Mateo, D., Zoss, B., Tokić, G., Wilhelm, E., Bouffanais, R., & al., et. (2017). Swarm-enabling technology for multi-robot systems. *Front Robot AI*. https://doi.org/10.3389/frobt.2017.00012

Couzin, I., Krause, J., Franks, N., & Levin, S. (2005). Effective leadership and decision-making in animal groups on the move. *Nature*. https://doi.org/10.1038/nature03236

Dias, P., Silva, M., Rocha Filho, G., Vargas, P., Cota, L., & Pessin, G. (2021). Swarm robotics: A perspective on the latest reviewed concepts and applications. *Sensors*. https://doi.org/10.3390/s21062062

Dorigo, M., Theraulaz, G., & Trianni, V. (2021). Swarm robotics: Past, present, and future. *Proc IEEE*. https://doi.org/10.1109/JPROC.2021.3072740

Erez, T., Tassa, Y., & Todorov, E. (2015). Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ode and physx. *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 4397–4404. https://doi.org/10.1109/ICRA.2015.7139807

Freeman, R., & Biro, D. (2009). Modelling group navigation: Dominance and democracy in homing pigeons. *J Navigation*. https://doi.org/10.1017/S0373463308005080

Helbing, F., D, & Vicsek, T. (2000). Simulating dynamical features of escape panic. *Nature*. https://doi.org/10.1038/35035023

Jadbabaie, A., Jie, L., & Morse, A. (2003). Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Trans Automat Contr*. https://doi.org/10.1109/TAC.2003.812781

Kaiser, T., Begemann, M., Plattenteich, T., Schilling, L., Schildbach, G., & Hamann, H. (2022). ROS2SWARM - a ROS 2 package for swarm robot behaviors. *International Conference on Robotics and Automation (ICRA)*. https://doi.org/10.1109/ICRA46639.2022.9812417

Leonard, N., & Fiorelli, E. (2001). Virtual leaders, artificial potentials and coordinated control of groups. *IEEE Conference on Decision and Control*. https://doi.org/10.1109/CDC.2001.980728

Mondada, F., Pettinaro, G. C., Guignard, A., Kwee, I. W., Floreano, D., Deneubourg, J.-L., Nolfi, S., Gambardella, L. M., & Dorigo, M. (2004). SWARM-BOT: A new distributed robotic concept. *Autonomous Robots*, *17*, 193–221. https://doi.org/10.1023/B:AURO.0000033972.50769.1c

Pinciroli, Carlo, Birattari, M., Tuci, E., Dorigo, M., Rey Zapatero, M. del, Vinko, T., & Izzo, D. (2008). Lattice formation in space for a swarm of pico satellites. *Ant Colony Optimization and Swarm Intelligence*. https://doi.org/10.1007/978-3-540-87527-7_36

Pinciroli, C., Lee-Brown, A., & Beltrame, G. (2015). Buzz: An extensible programming language for self-organizing heterogeneous robot swarms. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. https://doi.org/10.1109/IROS.2016.7759558

Pinciroli, Carlo, Trianni, V., O'Grady, R., Pini, G., Brutschy, A., Brambilla, M., Mathews, N., Ferrante, E., Di Caro, G., Ducatelle, F., & others. (2011). ARGoS: A modular, multi-engine simulator for heterogeneous swarm robotics. *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 5027–5034. https://doi.org/10.1109/IROS.2011.6094829

Reif, J. H., & Wang, H. (1999). Social potential fields: A distributed behavioral control for autonomous robots. *Robotics and Autonomous Systems*, *27*(3), 171–194. https://doi.org/10.1016/S0921-8890(99)00004-4

Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, 25–34. https://doi.org/10.1145/37402.37406

Spears, W., & Gordon, D. (1999). Using artificial physics to control agents. *International Conference on Information Intelligence and Systems*. https://doi.org/10.1109/ICIIS.1999.810278

Spears, W., Spears, D., Hamann, J., & Heil, R. (2004). Distributed, physics-based control of swarms of vehicles. *Autonomous Robots*. https://doi.org/10.1023/B:AURO.0000033970.96785.f2

Testa, A., Camisa, A., & Notarstefano, G. (2021). ChoiRbot: A ROS 2 toolbox for cooperative robotics. *IEEE Robot Autom Lett*. https://doi.org/10.1109/LRA.2021.3061366

Toner, J., & Tu, Y. (1998). Flocks, herds, and schools: A quantitative theory of flocking. *Physical Review E*, *58*(4), 4828. https://doi.org/10.1103/PhysRevE.58.4828

Trianni, V. (2008). *Evolutionary swarm robotics* (Vol. 108). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-77612-3

Vaughan, R. (2008). Massively multi-robot simulation in stage. *Swarm Intelligence*, *2*, 189–208. https://doi.org/10.1007/s11721-008-0014-4

Vicsek, T., Czirók, A., Ben-Jacob, E., Cohen, I., & Shochet, O. (1995). Novel type of phase transition in a system of self-driven particles. *Physical Review Letters*, *75*(6), 1226. https://doi.org/10.1103/PhysRevLett.75.1226

Zoss, B., Mateo, D., Kuan, Y., Tokić, G., Chamanbaz, M., Goh, L., & al., et. (2017). Distributed system of autonomous buoys for scalable deployment and monitoring of large waterbodies. *Auton Robot*. https://doi.org/10.1007/s10514-018-9702-0