Robots that Sync and Swarm: A Proof of Concept in ROS 2

Agata Gniewek, Michał Barciš and Christian Bettstetter

Abstract—A unified mathematical model for synchronisation and swarming has been proposed recently. Each system entity, called “swarmalator”, coordinates its internal phase and location with the other entities in a way that these two attributes are mutually coupled. This paper realises and studies, for the first time, the concept of swarmalators in technical systems. We adapt and extend the original model for its use on mobile robots and implement it in the Robot Operating System 2 (ROS 2). Simulations and experiments with small robots demonstrate the feasibility of the model and show its potential to be applied in real-world systems. All types of space-time patterns achieved in theory can be reproduced in practice. Applications can be found in monitoring, exploration, entertainment and art, among other domains.

Index Terms—Swarm robotics, synchronisation, swarmalators, ROS 2, pulse coupled oscillators, self-organisation.

I. INTRODUCTION

Two exciting phenomena of self-organisation occurring in natural and technical systems are synchronisation and swarming (see examples in [1]–[3]). In simple words, synchronisation is coordination in time; swarming is coordination in space. Scientific work on these two phenomena remained mostly disconnected until O’Keeffe, Hong and Strogatz [4] recently proposed a unified mathematical model for entities that both synchronise and swarm. The approach is to consider “oscillators whose phase dynamics and spatial dynamics are coupled” [4]. Such an entity is called swarmalator [4]. Its phase dynamics results from location-dependent synchronisation and its spatial dynamics results from phase-dependent aggregation. Groups of swarmalators show visually appealing spatio-temporal patterns, including different types of phase waves and cluster formations.

The general objective of our work in this domain is to transfer and adapt the swarmalator model to mobile robotics and employ it in real-world systems.

We see many applications for swarmalators in robotics. To give some examples, in monitoring and surveillance, the model can be used to coordinate robots during a patrolling mission or gather them around a point of interest. In exploration, a group of underwater vehicles can swim in a formation with the movements of their fins synchronised in order to improve performance. In art and entertainment, swarmalators can draw artificial paintings and perform aerial light shows with drones.

As a first step, we present this paper with the following contributions: we implement the swarmalator model in the Robot Operating System 2 (ROS 2), reproduce spatio-temporal patterns of [4], modify and extend the model to account for the specific movement properties of robots (namely collision avoidance and constraints in the movement directions), propose a new swarming platform based on Balboa robots [5] and finally use it to present a proof of concept demonstrating for the first time the feasibility of real robots acting as swarmalators.

The main reason we use ROS 2 instead of ROS is that it operates without a central unit, which makes it well suited for swarm robotics.

The paper is structured as follows: Section II presents a swarmalator model for robots. Section III describes the robot platform and the implementation in ROS 2. Section IV shows the results from simulations and experiments. Section V covers related work. Finally, Section VI concludes.

II. SYSTEM MODEL

A. The Original Model

A swarmalator \(i\) has a location \(\mathbf{x}_i \in \mathbb{R}^m\) in the \(m\)-dimensional space and an internal phase \(\phi_i \in [0,2\pi)\). Its natural frequency of oscillation is \(\omega_i\). The position difference between two entities \(i\) and \(j\) is \(\mathbf{x}_{ij} = \mathbf{x}_j - \mathbf{x}_i\) and their phase difference is \(\phi_{ij} = \phi_j - \phi_i\). The behaviour of entity \(i\) in a system of \(N\) entities is modelled by two differential equations [4]: Phase-dependent movement is given by

\[
\dot{x}_i = \frac{1}{N} \sum_{j \neq i} \left[ I_1(\mathbf{x}_{ij}) F(\phi_{ij}) - I_2(\mathbf{x}_{ij}) \right],
\]

and location-dependent synchronisation is given by

\[
\dot{\phi}_i = \omega_i + \frac{K}{N} \sum_{j \neq i} H(\phi_{ij}) G_\phi(\mathbf{x}_{ij}).
\]

The function \(I_1\) describes the spatial attraction between two entities and \(F\) determines how this attraction is influenced by their phase similarity. \(I_2\) describes the spatial repulsion between two entities. In a similar way, the synchronisation is defined by two functions: \(H\) is the attraction of the phases and \(G_\phi\) is its dependency on the proximity of two entities.

The following functions are used in [4]:

\[
I_1(\mathbf{x}_{ij}) = \frac{\mathbf{x}_{ij}}{||\mathbf{x}_{ij}||},
\]

\[
I_2(\mathbf{x}_{ij}) = \frac{\mathbf{x}_{ij}}{||\mathbf{x}_{ij}||^2},
\]

\[
F(\phi_{ij}) = 1 + J \cos \phi_{ij},
\]

\[
H(\phi_{ij}) = \sin \phi_{ij},
\]
\[ G_\phi(x_{ij}) = \frac{1}{\|x_{ij}\|}. \] (7)

The system has two parameters: \( J \) and \( K \). The parameter \( J \in [-1, 1] \) determines how strong the spatial attraction is influenced by phase similarity. If \( J \) is positive, an entity wants to be close to entities with the same phase. If it is negative, an entity is attracted by entities with the opposite phase. The original model does not account for the orientation of entities. The parameter \( K \in \mathbb{R} \) determines how strong the phases of two entities are coupled. Positive values of this coupling strength tend to lower the phase difference between entities. Negative values tend to increase it.

Applying this model on a set of entities yields after some time different patterns depending on the choice of parameters [4]. These patterns can be classified into five categories: static sync, static async, static phase wave, splintered phase waves, and active phase wave [4].

### B. A Model Suited for Robots

In order to apply the swarmalator model in robotics, we perform some modifications and extensions.

1) **Modifications due to Physical Constraints:** First, we need to take into account the specific characteristics of robots. For Balboa robots used in our work, the following movement limitations must be considered:

- **Movement constraints:** A swarmalator in the original model can move freely in all directions. In contrast, the robot can only turn around its centre or move in the direction it is facing, either forward or backward. It is therefore impossible to execute the velocity behaviour as described in Equation (1). Instead, we interpret it as the *desired velocity*, denoted by \( \mathbf{v}_i^{(d)} \) for robot \( i \). Additionally, we introduce a new state variable \( \theta_i \) for the orientation of the robot. Function \( S \) describes how the robot’s orientation is influenced by its desired velocity. Finally, the velocity \( \mathbf{x}_i \) is the component of the desired velocity in the direction that the robot is facing. All equations are given in Box 1.

- **Speed limitations:** The maximal speed of the robot is limited by its drivetrain. The linear and angular velocities obtained from the equations need to be appropriately limited.

- **Collision avoidance:** A swarmalator in the original model is treated as a point-shaped entity. In contrast, robots occupy a certain area. To prevent robots from colliding with each other, we define a circular safety area around each robot. The function \( I_2 \) is redefined using the distance between these safety areas, denoted by \( d_{ij} \). This ensures that the robots will repel each other if they get too close.

2) **Alignment of Orientations:** Second, we introduce an extension to the swarmalator model in which the orientations \( \theta \) can be aligned. The function \( R \) describes how the orientations of two entities are attracted. The function \( G_\phi \) determines how their spatial proximity influences this attraction. The strength of the attraction is controlled by the coefficient

\[
\lambda = \min \left\{ 1, \frac{\|\mathbf{v}_i^{(d)}\|}{PC} \right\}
\] (8)

with the parameter \( PC \in \mathbb{R}^+ \) defining the strength of attraction and the constant \( C \in \mathbb{R}^+ \) depending on the movement specifics. If \( PC = 0 \), the entity will always turn to the direction of its desired movement. As \( \lambda \) grows, the influence from other entities becomes higher and finally, for \( PC \to \infty \), the entity will be aligned only with its neighbours and never try to turn in a direction of the desired velocity.

The product \( PC \) can be seen as a threshold velocity above which an entity does not align with other entities but only with its desired velocity. A particular value is \( PC = 1 \) because the threshold velocity becomes equal to \( C \). In this paper, \( C \) is set to be the maximum possible velocity of the robots. If the model wants to make an entity move faster than possible, the orientation of other entities will not influence the entity at all but it will only turn in the desired direction. As soon as the desired velocity drops and is achievable, the orientation will start to be influenced by others. At the full stop, only the orientation attraction will be considered.

**Box 1: Swarmalator model for mobile robots**

\[
\mathbf{v}_i^{(d)} = \frac{1}{N} \sum_{j \neq i}^N [I_1(x_{ij})F(\phi_{ij}) - I_2(x_{ij})]
\]

\[
\dot{\theta}_i = (1 - \lambda) \frac{1}{N} \sum_{j \neq i}^N R(\theta_{ij})G_\phi(x_{ij}) + \lambda S(\theta_i, \mathbf{v}_i^{(d)})
\]

\[
\dot{x}_i = \mathbf{v}_i^{(d)} \cos \theta_i
\]

\[
\dot{\phi}_i = \omega_i + K \frac{1}{N} \sum_{j \neq i}^N H(\phi_{ij})G_\phi(x_{ij})
\]

The overall model is given in Box 1. Robots that synchronise and swarm according to this model are called *swarmalatorbots* in the following. We use \( I_1, F, H \) and \( G_\phi \) from the original model described above. The orientation attraction is chosen to be the same as the phase attraction:

\[
R(\theta_{ij}) = \sin \theta_{ij}, \quad (9)
\]

\[
G_\phi(x_{ij}) = \frac{1}{\|x_{ij}\|}. \quad (10)
\]

Furthermore, we use:

\[
I_2(x_{ij}) = \frac{x_{ij}}{d_{ij}}, \quad (11)
\]

\[
S(\theta_i, \mathbf{v}_i^{(d)}) = \sin(\angle(\mathbf{v}_i^{(d)}) - \theta_i). \quad (12)
\]

### III. IMPLEMENTATION

This section describes the robot platform used, the implementation of the swarmalatorbots in ROS2 and other modules required to perform experiments.
A. Mobile Robot Platform

Our requirements for a robot platform are as follows:
- the price should be in the order of 200 EUR;
- the computational power should be high enough to perform tasks such as localisation and basic computer vision;
- the operating system should be Linux;
- hardware extensions with additional sensors and communication interfaces should be possible;
- the assembly time should not exceed two person-hours;
- the platform should also be suited for other research activities in our group.

We have identified multiple platforms that fulfil these requirements and eventually selected the Balboa [5] robots from Pololu. One of its advantages for our purpose is a good state estimation capability due to an integrated inertial measurement unit (IMU) and quadrature encoders connected to the motors. Its computational power is provided by a RaspberryPi 3B+ computer. Additionally, its self-balancing behaviour is visually attractive, especially when multiple robots operate next to each other.

The official software library for the low level controller (LLC) provided by the vendor has been tuned to our needs. We changed the communication interface between the computing board and the LLC to a serial connection (using UART protocol). This enables us to remotely reprogram the microcontroller and implement a protocol for robot control and readout of measurements. In addition, we have tuned the controller responsible for balancing to allow it to work with additional load. All the modifications are published in the GitLab repository [10]. Finally, for the visualisation of the phase of a swarmalatorbot, we have attached a strip of Neopixel LEDs to each robot.

B. Swarmalatorbots in ROS 2

The swarmalatorbots are implemented as nodes in ROS 2. They periodically publish their states to all other entities with a configurable frequency. Each swarmalatorbot gathers the received messages and calculates its new control variables from the model (Box 1). The implementation of the swarmalatorbot can run in the simulation environment as well as on the robots in the experiment.

In order to make the development faster and results easier to present, we have implemented a module for visualisation. It listens to messages passed between swarmalatorbots and renders images (see Figure 1). It enables us to visualise both the simulation and the real robots using the same software.

Fig. 1: A picture rendered in a simulation of swarmalators.

1For the simulation of more than 20 entities the message passing is substituted with intra-process communication, to speed up computations.

C. Localisation

Swarmalatorbots need to know their positions in a global reference frame. Outdoors they could acquire their location with GPS, but as our proof of concept operates indoors, they use a motion capture system (OptiTrack [7]). For the simulation purposes a position is calculated based on a physical movement model by an external node.

D. Wireless Communication

A swarmalatorbot has to exchange its state information with all other swarmalatorbots and also receive information about its location from OptiTrack.

Standard WiFi is used for both tasks due to its ease of configuration, availability and high throughput. Although the wireless modules on RaspberryPi support 802.11ac, we use by default the 802.11bg standard, for the sake of compatibility with other devices. The ad-hoc mode is used instead of the infrastructure mode to make the system design robust against failure or loss of some nodes. Although 802.11 is not optimised for small packets [8] used by our application, its performance is sufficient for the experiments.

E. Software Deployment for Multi-Robot Systems

An automated deployment process is an important part of software ecosystems. Such solutions are very beneficial especially in swarm robotics since setting up each one of tens of robots individually is a time-consuming task.

As there is no solution that fulfils our requirements, we have developed a custom framework based on Ansible [9]. It enables us to remotely install, start and stop a program on a selected subset of robots at the same time. The deployment can be made independently of the robot platform and programming language. It is possible to specify robot groups (e.g. depending on hardware configuration) on which different versions of software are deployed. The solution is published in the GitLab repository [10].

IV. Resulting Patterns

We now study the behaviour of $N$ swarmalatorbots via ROS 2 simulations and experiments with our robotic platform. After initial placement, each entity acts as a swarmalator and interacts with all other entities. As time progresses, each entity changes its state in terms of position $x$, orientation $\theta$ and phase $\phi$. After some time, a snapshot of the resulting pattern is taken. The orientation of an entity is visualised by an arrow; its phase is shown by a colour. All tests are done with the natural frequency $\omega_i = 0 \forall i$.

A. Simulation Results

Simulations are conducted for $N = 100$ entities, whose initial placement is sampled from a uniform random distribution in a square area with $x, y \in [-1, 1]$ length units. As a first step, we perform simulations of the original swarmalator model for the purpose of mutual validation. All five patterns introduced in [4] are qualitatively reproduced.

Next, we simulate the model suited for robots in order to check whether the modifications due to physical constraints
influence the capability to form patterns. Figure 2 shows some examples of patterns achieved without alignment of orientations ($P=0$). We conclude that it is possible to obtain all the patterns in this robot model as well.

For the parameter set $(J, K) = (0.1, 1)$, the model yields a pattern in which all entities are synchronised and regularly distributed inside a circle. The set $(J, K) = (0.1, -1)$ gives a similar spatial pattern but keeps the nodes in asynchrony. For $(J, K) = (1, 0)$, all entities are regularly distributed on a ring, where the ones with similar phases are gathered together. The last two states are non-stationary. For $(J, K) = (1, -0.1)$, the entities create disconnected clusters of similar phases around a ring; they keep moving inside those groups; this pattern is called splintered phase wave [4]. For $(J, K) = (1, -0.3)$, the entities move around the ring while their phases oscillate, which is called active phase wave [4].

The model also works fine if we include orientation alignment ($P > 0$). Figure 3 shows examples for the three static patterns. The two non-stationary patterns can, in general, not be formed with our orientation alignment.

B. Results with Robotic Prototype

Experiments are made with $N = 10$ robots running the model implemented in ROS 2. Example results are presented in Figures 4 and 5. All patterns except the splintered phase wave are reproduced. The splintered phase wave is not visible in our setting due to the low number of entities. For all stationary patterns (Figure 4), both versions with and without orientation alignment are created. Despite the lower number of entities, the formed patterns are clearly visible and can be mapped to the different pattern types. Figure 4 contains four figures for each stationary state: the bird’s-eye view of the robots forming the pattern (Robots), snapshot from the visualisation tool (Visualisation), trace with the formed pattern, showing that the state is stationary (Trace — formed pattern) and the trace of pattern forming (Trace — pattern forming). Older samples are visualised with lighter colour. The active phase wave formed by robots is presented in Figure 5a. The trace (Figure 5b) shows the robots’ movement after the pattern was formed.

A video about our experiments can be seen at the Website [11].

V. RELATED WORK

Related work is in the application of algorithms for self-organisation in robotics and in platforms for swarm robotics.

A. Self-Organisation Applications in Robotic Swarms

The swarmalator model was introduced by O’Keeffe et al. in [4] and analysed in more detail in [12]. In both cases, the size of the entities and possible movement constraints are neglected. As mentioned, these properties need to be taken into account in robotics.

There are different methods for pattern formation and control (see [13]–[15]). Self-organised flocking of mobile robots was studied and evaluated in simulations and on robots by Turgut et al. [16] and Ferrante et al. [17]. Robotic swarm navigation with obstacle avoidance was analysed in [18].

Synchronisation is often applied in technology, e.g. for slotted communication or in robotic swarms to allocate and
Fig. 4: Stationary patterns formed by swarmalatorbots.
execute tasks together. One of the popular algorithms applied in those scenarios is firefly synchronisation because entities have to exchange only simple pulses. It can be used in wireless networks [19], [20], but other, simpler communication interfaces can be used to emit and receive pulses as well (e.g., sound [21] and light [22]).

There are a few attempts to use the synchronisation pulses also for swarm coordination. Christensen et al. proposed a method in which robots detect faulty agents that stopped blinking and fix them or take their tasks over [23]. Hartbauer et al. described how light emitted by robots can be used as a guiding signal [24].

B. Platform for Swarm Robotics

There are multiple robotic platforms commonly used for swarm applications, but most of them have very limited computational power (e.g., Kilobots [25] and Spiderinos [26]). For this work, we decided to prepare a swarming platform capable of executing more complex tasks, which is able to run ROS. The official ROS platforms (e.g., Turtlebot [27]) are often more expensive than Balboa.

VI. CONCLUSIONS AND OUTLOOK

The swarmalator concept for coupled synchronisation and swarming has been realised and investigated for the first time in a real-world system. We have adapted and extended the original mathematical model for its use in mobile robotics and implemented it in ROS 2. Simulations and experiments on our Balboa-based platform serve as a proof of concept. Applications can be found in fields with coordinated mobility of multiple entities, such as monitoring, exploration and entertainment. The colourised space-phase traces can be considered as robotic art.

We observed in our experiments that the communication delay has huge impact on the patterns stationarity if \( \omega > 0 \). For example, the static phase wave starts to rotate. This phenomenon requires further investigation. From a more general point of view, future work will consider swarmalorbots in three dimensions using a small swarm of commercial drones.

ACKNOWLEDGEMENTS

This work was supported by the Karl Popper Kolleg on Networked Autonomous Aerial Vehicles at the University of Klagenfurt and by the Austrian Science Fund (FWF) under grant P30012-NBL on self-organising synchronisation.

REFERENCES

[1] S. Camazine, J.-L. Deneubourg, N. R. Franks, J. Sneyd, G. Theraula, and E. Bonabeau, Self-Organization in Biological Systems, 1st ed. Princeton, NJ: Woodstock: Princeton University Press, Sep. 2003.
[2] S. H. Strogatz, Sync: How Order Emerges From Chaos In the Universe, Nature, and Daily Life. New York, NY: Hachette Books, Apr. 2004.
[3] H. Hamann, Swarm Robotics: A Formal Approach. Springer, 2018.
[4] K. P. O’Keeffe, H. Hong, and S. H. Strogatz, “Oscillators that sync and swarm,” Nature Communications, vol. 8, no. 1, Nov. 2017.
[5] “Pololu Balboa 32U4 Balancing Robot user’s guide,” https://www.pololu.com/docs/0J370.
[6] “Balboa low level controller source code,” https://gitlab.aau.at/aau-nav/development/balboa_likc-aws.
[7] “OptiTrack - motion capture systems,” http://optitrack.com/
[8] J. Jun, P. Peddabachagadi, and M. Sichitiu, “Theoretical maximum throughput of IEEE 802.11 and its applications,” in Proc. IEEE Intern. Symp. on Network Comput. and Applications, May 2003, pp. 249–256.
[9] L. Hochstein and R. Moser, Anisible: Up and Running: Automating Configuration Management and Deploying the Easy Way. O’Reilly, Jul. 2017.
[10] “Source code repository with Ansible playbooks,” https://gitlab.aau.at/aau-nav/development/ansible-playbooks.
[11] “Video: Robots that sync and swarm: A proof of concept in ROS2,” https://bettstetter.com/swarmalatorbots/.
[12] K. P. O’Keeffe, J. H. M. Evers, and T. Kolokolnikov, “Ring states in swarmalator systems,” Physical Review E, vol. 98, no. 2, p. 022203, Aug. 2018.
[13] I. Suzuki and M. Yamashita, “Distributed anonymous mobile robots: Formation of geometric patterns,” SIAM Journal on Computing, vol. 28, no. 4, pp. 1347–1363, Jan. 1999.
[14] L. Barnes, M. Fields, and K. Valavanis, “Unmanned ground vehicle swarm formation control using potential fields,” in Proc. Mediterranean Conf. on Control Automation, Jun. 2007.
[15] H. C.-H. Hsu and A. Liu, “Multiagent-based multi-team formation control for mobile robots,” Journal of Intelligent and Robotic Systems, vol. 42, no. 4, pp. 337–360, Apr. 2005.
[16] A. E. Turgut, H. Çelikkanat, F. Gökcə, and E. Şahin, “Self-organized flocking in mobile robot swarms,” Swarm Intelligence, vol. 2, no. 2-4, pp. 97–120, Dec. 2008.
[17] E. Ferrante, A. E. Turgut, C. Huepe, A. Stranieri, C. Pincirolli, and M. Dorigo, “Self-organized flocking with a mobile robot swarm: A novel motion control method,” Adaptive Behavior, vol. 20, no. 6, pp. 460–477, Dec. 2012.
[18] L. S. Junior and N. Nedjah, “Efficient strategy for collective navigation control in swarm robotics,” Procedia Computer Science, vol. 80, pp. 814–823, Jan. 2016.
[19] G. Brandner, U. Schilcher, and C. Bettstetter, “Firefly synchronization with phase rate equalization and its experimental analysis in wireless systems,” Computer Networks, vol. 97, pp. 74–87, Mar. 2016.
[20] Y.-W. Hong and A. Scaglione, “A scalable synchronization protocol for large scale sensor networks and its applications,” IEEE Journal on Selected Areas in Communications, vol. 23, no. 5, pp. 1085–1099, May 2005.
[21] V. Trianni and S. Nolfi, “Self-organizing sync in a robotic swarm: A dynamical system view,” IEEE Transactions on Evolutionary Computation, vol. 13, no. 4, pp. 722–741, Aug. 2009.
[22] F. Perez-Diaz, R. Zillmer, and R. Gross, “Firefly-inspired synchronization in swarms of mobile agents,” in Proc. Intern. Conf. on Autonomous Agents and Multiagent Syst., Richland, SC, 2015, pp. 279–286.
[23] A. L. Christensen, R. O’Grady, and M. Dorigo, “From fireflies to fault-tolerant swarms of robots,” IEEE Transactions on Evolutionary Computation, vol. 13, no. 4, pp. 754–766, Aug. 2009.
[24] M. Hartbauer and H. Römer, “A novel distributed swarm control strategy based on coupled signal oscillators,” Bioinspiration & Biomimetics, vol. 2, no. 3, pp. 42–55, 2007.
[25] M. Rubenstein, C. Ahler, and R. Nagpal, “Kilobot: A low cost scalable robot system for collective behaviors,” in Proc. IEEE Intern. Conf. on Robotics and Automation (ICRA), May 2012, pp. 3293–3298.
[26] F. Perez-Diaz, S. Zhevchyk, F. Steinkellner, and W. Elmenreich, “Spiderino-A low-cost robot for swarm research and educational purposes,” in Proc. Workshop on Intelligent Solutions in Embedded Systems (WISES), Jun. 2017, pp. 35–39.
[27] “TurtleBot,” https://www.turtlebot.com/.

Fig. 5: Active phase wave formed by swarmalorbots.