Navigation without localization using stable cycles

Quentin Brateau, Fabrice Le Bars, Luc Jaulin

▶ **To cite this version:**

Quentin Brateau, Fabrice Le Bars, Luc Jaulin. Navigation without localization using stable cycles. 2024. hal-04846273

HAL Id: hal-04846273
https://hal.science/hal-04846273v1

Preprint submitted on 18 Dec 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution - NonCommercial - ShareAlike 4.0 International License

Navigation without localization using stable cycles

Quentin Brateau
Lab-STICC UMR 6285, Robex team
ENSTA Bretagne
Brest, France
quentin.brateau@ensta-bretagne.org

Fabrice Le Bars
Lab-STICC UMR 6285, Robex team
ENSTA Bretagne
Brest, France
fabrice.le_bars@ensta-bretagne.fr

Luc Jaulin
Lab-STICC UMR 6285, Robex team
ENSTA Bretagne
Brest, France
lucjaulin@gmail.com

Abstract—In a Global Navigation Satellite System (GNSS) denied environment, it is difficult to use classical navigation methods as the position of the robot is unknown. This paper presents a robot control method using cycles. The cycle paradigm first proposes to control the robot by making it follow a cycle while its position remains unknown. This cycle is then moved and stabilized using a few measurements in the environment which is assumed to be known. Once the cycle is stable, the position of the robot is easier to estimate. This makes it possible to navigate frugally and stealthily, without getting lost.

Index Terms—Marine Robotics, Control, Stability

I. Introduction

Classical methods for navigation rely on control laws based on an estimation of the robot position. This position estimate is derived from exteroceptive sensors such as GNSS receivers or acoustic positioning solutions in marine robotics [1]. However, these sensor measurements may not always be accessible, especially in GNSS-denied environments, or in sensitive environments where acoustic positioning systems cannot be deployed. In these cases, the robot must rely on proprioceptive sensors for state estimation through dead-reckoning.

A prevalent trend in enhancing the robustness of systems involves an increase in the number of sensors to collect more data. While this strategy adds complexity to system design and may requires more computing power, it can also reduce the stealthiness of robots. Conversely, there is a rising interest in frugal approaches within robotics that advocate for using minimal information and computational power to control robots [2].

Additionally, bio-mimetism is gaining traction in robotics. Biomimicry provides elegant and efficient solutions in robotics. For example, control laws inspired by bees may improve state estimation for flying robots using optical flow based visual odometry [3]. For underwater navigation, some marine animals can navigate long distances without any position estimation. They rely on proprioceptive and some exteroceptive sensors to perform cycles through the seasons. That is the case for migratory birds or sea turtles [4].

This work has been supported by the French Government Defense procurement and technology agency (AID).

A video of experiments is available at https://teusner.github.io/projects/icra2025.

The goal of this work is to draw inspiration from these methods and to develop a method that allows a robot to navigate in a zone without getting lost, and without external positionning system. The proposed approach is to control the robot along cycles. This cycle described by the trajectory of the robot is then moved in the environment toward a stable cycle using a few exteroceptive measurements. By using cycle navigation, the robot can operate with limited information while maintaining stealth. This method is also close to frugal approaches as it uses minimal information and computational power.

Using stable cycles for controlling and localizing robotic systems represents a new paradigm. The objective of this work is to formalize the application of cycles in robotics, demonstrate their stability, and showcase their practical implementation. In this work, control modules for using stable cycles in vehicle navigation will be built iteratively.

II. Dynamical system

A. Evolution function

Consider a system with a state $\mathbf{x} \in \mathcal{S}$, a constant input $\mathbf{u} \in \mathcal{U}$, and governed by the continuous evolution equation (1) [5]–[7].

$$\dot{\mathbf{x}}(t) = \mathbf{f}_{\mathbf{u}}(\mathbf{x}(t)) \tag{1}$$

There exists a flow function $\boldsymbol{\varphi}_{\mathbf{u}} : \mathcal{S} \times \mathcal{T} \to \mathcal{S}$ which is the solution of the differential equation (1) with initial condition $\mathbf{x}_0 \in \mathcal{S}$ [7], satisfying properties of Equation 2.

$$\begin{aligned} &\forall (\mathbf{x}_0, t1, t2) \in \mathcal{S} \times \mathcal{T}^2 \\ &\qquad \boldsymbol{\varphi}_{\mathbf{u}}(\mathbf{x}_0, t_1) = \mathbf{x}(t_1) \\ &\qquad \boldsymbol{\varphi}_{\mathbf{u}}(\mathbf{x}_0, 0) = \mathbf{x}_0 \\ &\qquad \boldsymbol{\varphi}_{\mathbf{u}}(\boldsymbol{\varphi}_{\mathbf{u}}(\mathbf{x}_0, t1), t2) = \boldsymbol{\varphi}_{\mathbf{u}}(\mathbf{x}_0, t1 + t2) \end{aligned} \tag{2}$$

Note that the analytical expression of the flow function $\boldsymbol{\varphi}_{\mathbf{u}}$ is not always available. However, an approximation of this function can be computed by numerical integration of the differential equation (1).

B. Cyclic Timed Automata

The system can be controlled by a timed automata as defined in [8], [9]. A timed automata is a finite state machine extended with a finite collection of real-valued clock variables controlling the transitions between states.

Fig. 1 shows an example of cyclic timed automata, which is a determinist automata, with only one transition in and one transition out of each state, and which has a cycle shape.

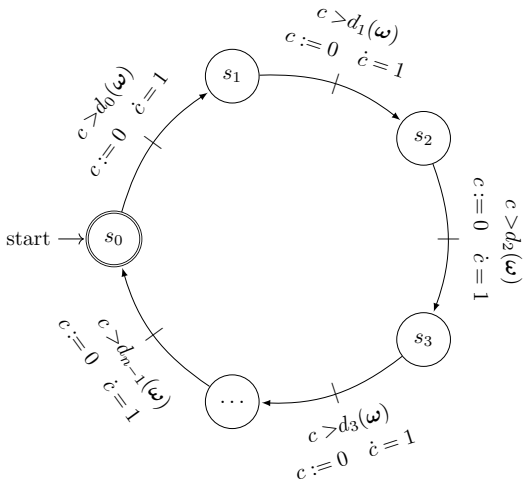


Fig. 1: Timed automata

In this automata, a clock c is continuously ticking and is reset to zero after each transition. Transitions between states s_i occurs when the clock c exceeds a duration $d_i(\boldsymbol{\omega})$, where $\boldsymbol{\omega}$ is a parameter vector introduced to control the duration of each state.

A constant system input $\mathbf{u}$ is associated to each state. Consequently, the system follows a different evolution function $\mathbf{f}_{\mathbf{u},i}$ for each state of the timed automata, and there exists a flow function $\boldsymbol{\varphi}_{\mathbf{u},i}$ associated to each state s_i. Introducing the operator $\bigcirc_{i=0}^{n}$ to denote the composition of functions, $\boldsymbol{\phi}$ represents the flow functions over a complete iteration of the automata. Figure 2 shows the composition of flow functions over a cycle.

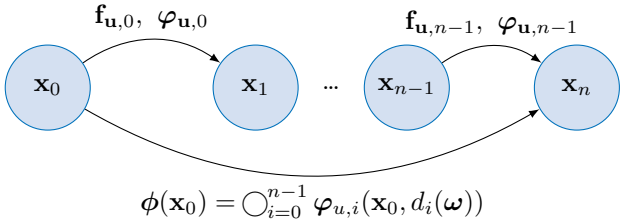


Fig. 2: Composition of flow functions over a cycle

C. Synchronization condition

The duration of one iteration of the timed automata is denoted by $T(\boldsymbol{\omega})$ and is called the cyclic period. The cyclic period respects Equation (3).

$$T(\boldsymbol{\omega}) = \sum_{i=0}^{n-1} d_i(\boldsymbol{\omega}) \tag{3}$$

The dynamical system and the timed automata are synchronized if they meet the condition of Equation (4). Equation (4) implies that after one iteration the system comes back to the same state.

$$\boldsymbol{\phi}(\mathbf{x}(t)) \triangleq \mathbf{x}(t + T(\boldsymbol{\omega})) = \mathbf{x}(t) \tag{4}$$

The block diagram shown in Fig. 3 summarizes the control architecture at this point, where $\mathcal{A}$ represent the automata and the system is represented by its evolution equation.

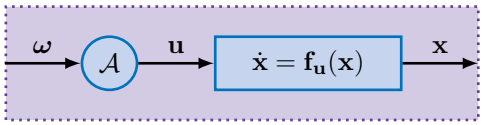


Fig. 3: Block diagram of the robot controlled by a timed automata

As an example, consider a vehicle following the kinematic model of the Dubins car [10] presented in Equation (5).

$$\dot{\mathbf{x}} = \begin{cases} \dot{x} = v \cdot \cos(\theta) \\ \dot{y} = v \cdot \sin(\theta) \\ \dot{\theta} = u \end{cases} \tag{5}$$

In Equation (5), $\mathbf{x} = \begin{bmatrix} x & y & \theta \end{bmatrix}^T$ is the pose of the robot, v is its speed supposed constant and positive, and u, the input, is the turning rate. Hence, the trajectory of the vehicle can follow straight lines when $u = 0$, and circle arcs when $u \neq 0$.

The cyclic timed automata shown in Fig. 4 is designed to control the trajectory of the vehicle such that the system is following a square cycle as shown in Fig. 5, by alternating straight lines and circle arcs. As after one iteration of the automata with a zero input $\boldsymbol{\omega}$ the vehicle comes back to its initial state, the vehicle and the automata are synchronized.

III. Cycles abstraction

A. Discretization

Moving up a level of abstraction, the cycle is now considered as the system to control. The cycle is evolving in the plane and the robot is still following the cycle.

By denoting by $\boldsymbol{\eta}_k$ the state of the cycle and $\boldsymbol{\omega}_k$ the input of the cycle at the beginning of the k^{th} iteration, the cycle is modeled by Equation (6).

$$\boldsymbol{\eta}_k = \bigcirc_{i=0}^{k-1} \boldsymbol{\phi}(\mathbf{x}, T(\boldsymbol{\omega}_i)) \tag{6}$$

With the cycle abstraction, the block diagram of Fig. 3 is simplified as in Fig. 6.

Note that with a non-zero input $\boldsymbol{\omega}_k$, the cycle is destabilized and no longer meets the synchronization

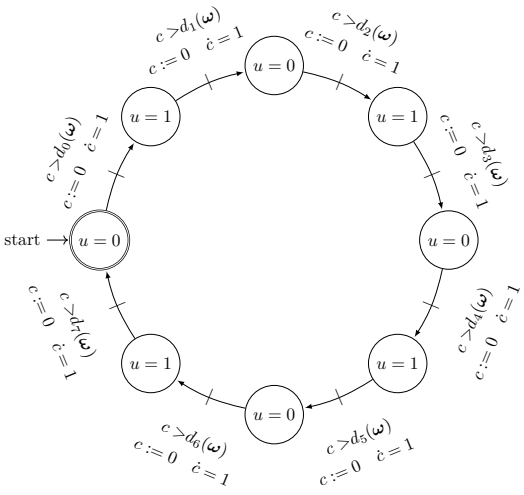


Fig. 4: Square cyclic timed automata

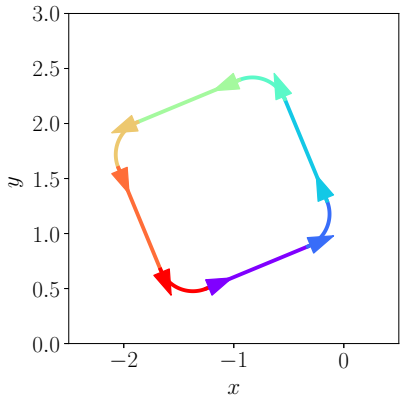


Fig. 5: Square cycle described by the robot trajectory

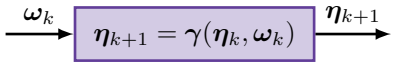


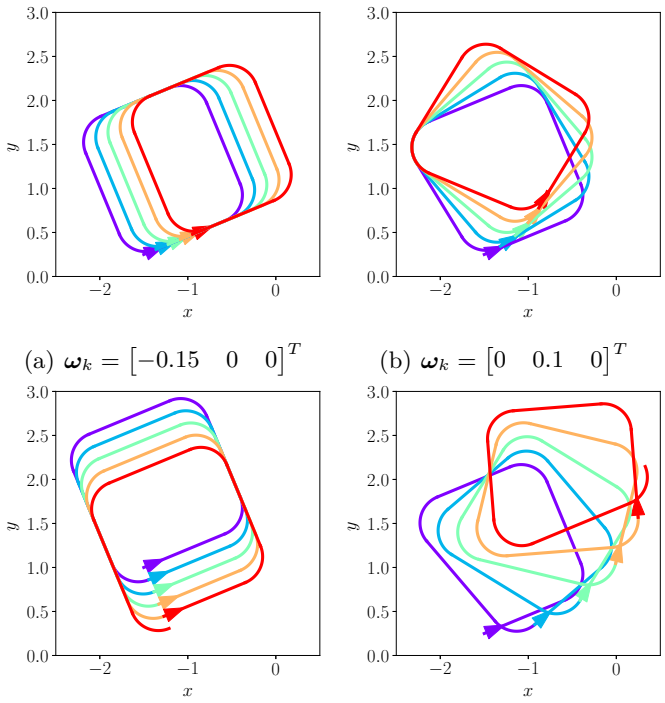
Fig. 6: Block diagram of the controlled cycle

condition of Equation (4). Yet, when the input is set back to zero, the system goes through regular and undisturbed cycles again.

For the square cycle example, the state of the cycle is $\boldsymbol{\eta}_k = \begin{bmatrix} x_k & y_k & \theta_k \end{bmatrix}^T$ and corresponds to the pose of the vehicle when starting the k^{th} iteration of the cyclic timed automaton.

As the cycle state has three degrees of freedom, $\boldsymbol{\omega}_k = \begin{bmatrix} \omega_{k,0} & \omega_{k,1} & \omega_{k,2} \end{bmatrix}^T$ is the three dimensional input of the system. The duration of states s_4, s_5, and s_6 will be adjusted by adding respectively $\boldsymbol{\omega}_k, 0$, and $\boldsymbol{\omega}_{k,1}$, and $\boldsymbol{\omega_{k,2}}$. Note that the inputs were selected to control all three degrees of freedom of the cycle while avoiding redundancy.

Fig. 7 shows the effect of an input on the cycle state over five iterations. Figs. 7a, 7b, and 7c show respectively the effect of inputs $\omega_{k,0}$, $\omega_{k,1}$, and $\omega_{k,2}$, and Fig. 7d shows the effect on the cycle of a complete input $\boldsymbol{\omega} = \begin{bmatrix} 0.15 & -0.2 & -0.2 \end{bmatrix}^T$. The initial pose of the cycle is then controllable.



(a) $\boldsymbol{\omega}_k = \begin{bmatrix} -0.15 & 0 & 0 \end{bmatrix}^T$ (b) $\boldsymbol{\omega}_k = \begin{bmatrix} 0 & 0.1 & 0 \end{bmatrix}^T$

(c) $\boldsymbol{\omega}_k = \begin{bmatrix} 0 & 0 & 0.15 \end{bmatrix}^T$ (d) $\boldsymbol{\omega}_k = \begin{bmatrix} -0.1 & 0.2 & 0.2 \end{bmatrix}^T$

Fig. 7: Cycle control under different inputs

Equation (7) models the evolution of this system for the square cycle example.

$$\boldsymbol{\gamma}(\boldsymbol{\eta}_k, \boldsymbol{\omega}_k) = \boldsymbol{\eta}_k + \begin{bmatrix} -cos(\theta_k) & -sin(\theta_k) & 0 \\ 0 & 0 & 1 \\ sin(\theta_k) & -cos(\theta_k) & 0 \end{bmatrix} \cdot \boldsymbol{\omega}_k \quad (7)$$

B. World frame control

A change of input lets the cycle be controlled in the world frame instead of in the cycle frame. Denoting by $\boldsymbol{\nu}_k$ the requested displacement of the cycle in the world frame, $\boldsymbol{\omega}_k$ equals $\boldsymbol{\zeta}(\theta_k, \boldsymbol{\nu}_k)$ defined by (8) [1]. Equation (8) acts as a feedback linearization on the system [5], [6].

$$\boldsymbol{\zeta}(\theta_k, \boldsymbol{\nu}_k) = \begin{bmatrix} -cos(\theta_k) & 0 & sin(\theta_k) \\ -sin(\theta_k) & 0 & -cos(\theta_k) \\ 0 & 1 & 0 \end{bmatrix} \cdot \boldsymbol{\nu}_k \quad (8)$$

Therefore, the system is now described by the linear evolution equation (9) and the block diagram of the regulated cycle is shown in Fig. 8.

$$\boldsymbol{\eta}_{k+1} = \boldsymbol{\eta}_k + \boldsymbol{\nu}_k \quad (9)$$

[1] θ_k can be measured with a compass or estimated

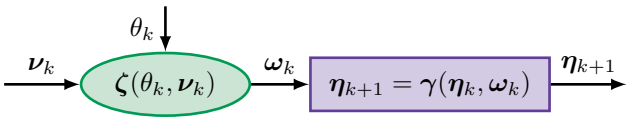
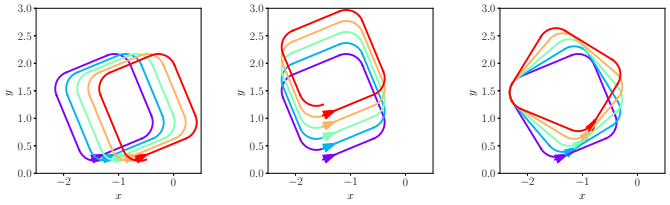


Fig. 8: Block diagram of the controlled cycle

In Fig. 9, a constant input $\boldsymbol{\nu}_{k,0}$, $\boldsymbol{\nu}_{k,1}$, and $\boldsymbol{\nu}_{k,2}$ is applied over 5 iterations on the system and cycles are well moved in the world frame.



(a) $\boldsymbol{\nu}_{k,0} = 0.2$ (b) $\boldsymbol{\nu}_{k,1} = 0.2$ (c) $\boldsymbol{\nu}_{k,0} = 0.1$

Fig. 9: Cycle control in the world frame

C. Adding Measurements

A set of measurements $\boldsymbol{\mu}_k$ is now introduced along the cycle. These measurements are used to close the loop and to control the system toward a desired state. This system is described by Equation (10).

$$\mathcal{S} : \begin{cases} \boldsymbol{\eta}_{k+1} = \boldsymbol{\gamma}(\boldsymbol{\eta}_k, \boldsymbol{\omega}_k) \\ \boldsymbol{\mu}_k = \boldsymbol{\sigma}(\boldsymbol{\eta}_k, \boldsymbol{\omega}_k) \end{cases} \tag{10}$$

In the square cycle example, Equation (11) simulates a depth-ranging sonar at the robot pose $\mathbf{x}$. The simulated seafloor is shown in Fig. 10a.

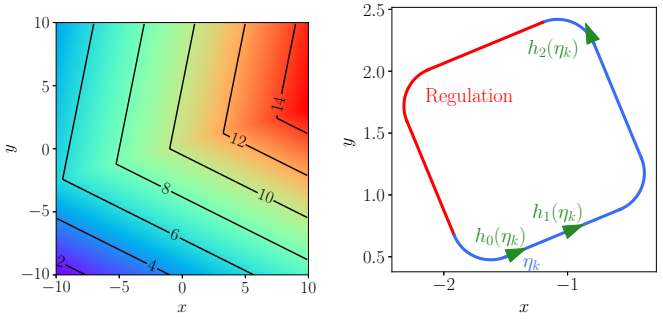
$$\begin{cases} g(\mathbf{x}) = 10 + \min\limits_{i=0,1} \left\{ 0.1 \cdot \det(\mathbf{b_i} - \mathbf{a_i}, \begin{bmatrix} x & y \end{bmatrix}^T - \mathbf{a_i}) \right\} \\ a_0 = \begin{bmatrix} 0 & 5 \end{bmatrix}^T, \quad b_0 = \begin{bmatrix} -1 & 0 \end{bmatrix}^T \\ a_1 = \begin{bmatrix} -1 & 0 \end{bmatrix}^T, \quad b_1 = \begin{bmatrix} 5 & -3 \end{bmatrix}^T \end{cases} \tag{11}$$

Three measurements are taken along the cycle, as shown in Fig. 10b, one at the beginning position, one in the middle of the first straight line, and one at the end of the second straight line. The function $\mathbf{h}_j$, which results from the composition of flow functions $\boldsymbol{\varphi}_{u,i}$, gives the measurement positions [2]

D. Regulation

A regulator ensures convergence of the state of the system toward the reference $\bar{\boldsymbol{\eta}}$ determined by $\bar{\boldsymbol{\mu}}$. This regulator is a simple proportional corrector that moves and rotates the cycle depending on measurements of the seafloor shown in Fig. 10a.

[2]Note that $\mathbf{h}_j$ only depends on the state of the cycle $\boldsymbol{\eta}_k$ and does not depend on the input $\boldsymbol{\omega}_k$ as the regulated part of the cycle is not affecting the measurements positions as shown in Fig. 10b.



(a) Simulated seafloor (b) Measurement positions

Fig. 10: Measurements environment and setup

Equation (12) defines the regulator, and coefficients k_i are chosen to stabilize the system. k_1 modulates the cycle position in the x direction, k_2 in the y direction, and k_3 modulates its orientation.

$$\boldsymbol{\lambda}(\boldsymbol{\mu}_k, \bar{\boldsymbol{\mu}}) = \begin{bmatrix} k_1 & 0 & 0 \\ 0 & 0 & k_2 \\ -k_3 & k_3 & 0 \end{bmatrix} \cdot (\bar{\boldsymbol{\mu}} - \boldsymbol{\mu}_k) \tag{12}$$

The cycle is now controlled and regulated in the plane to reach the target state $\bar{\boldsymbol{\eta}}$ specified by $\bar{\boldsymbol{\mu}}$. Fig. 11 shows the block diagram of the regulated system.

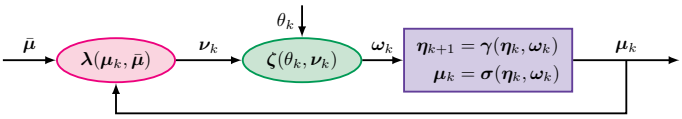


Fig. 11: Block diagram of the autonomous system

From this block diagram is derived the equation of the autonomous system given by Equation (13).

$$\boldsymbol{\eta}_{k+1} = \boldsymbol{\xi}_{\bar{\boldsymbol{\mu}}}(\boldsymbol{\eta}_k) \tag{13}$$

Equation 13 is linearized around $\bar{\boldsymbol{\eta}}$ such that it can be written as Equation (14).

$$\boldsymbol{\eta}_{k+1} = \mathbf{A} \cdot (\bar{\boldsymbol{\eta}} - \boldsymbol{\eta}_k) \tag{14}$$

By expressing eigenvalues of $\mathbf{A}$ relative to coefficients k_i, conditions of stability of the regulator are derived. For a discrete system, eigenvalues have to belong to the unit circle to ensure stability. Here eigenvalues e_i are given by Equation (15).

$$\begin{cases} e_0 = 1 - 0.025k_0 - 0.03k_1 + 0.042\sqrt{0.35k_0^2 - k_0k_1 + \dfrac{k_1^2}{2}} \\ e_1 = 1 - 0.025k_0 - 0.03k_1 - 0.042\sqrt{0.35k_0^2 - k_0k_1 + \dfrac{k_1^2}{2}} \\ e_2 = 1 - 0.05 \cdot k_2 \end{cases} \tag{15}$$

By tuning values of the regulator to $k_0 = 1$, $k_1 = 1$, and $k_2 = 1$, the system is stable as all eigenvalues are in the unit circle [3].

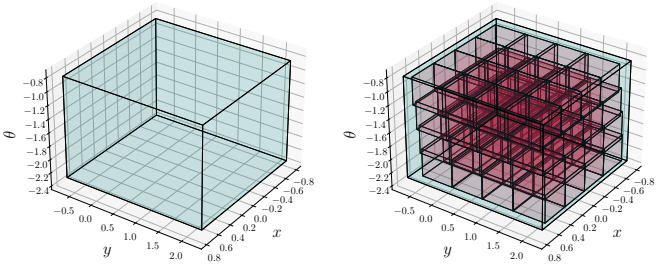
E. Cycle stability

The cycle stability is proven using an interval analysis approach. The proposed approach is to find a positive invariant set of state $\mathbb{P}$ [11], [12] for the autonomous system. This set must satisfy the condition in Equation (16), which means that as soon as the state of the system enters the set $\mathbb{P}$, it is forever captured in it.

$$\boldsymbol{\xi}_{\bar{\boldsymbol{\mu}}}(\mathbb{P}) \subset \mathbb{P} \tag{16}$$

$\mathbb{P} = \begin{bmatrix}[-0.75, 0.75] & [-0.75, 2.25] & [-2.32, -0.82]\end{bmatrix}$ is a positive invariant set for the square example. This box shown in blue in Fig. 12a is split into $n_{split} = 4$ in each dimension to reduce the wrapping effect of interval analysis [13]. The evolution function $\boldsymbol{\xi}_{\bar{\boldsymbol{\mu}}}$ is then applied to each box once, and the resulting boxes are plot in pink on Fig. 12b.

As all resulting boxes are a subset of $\mathbb{P}$, $\mathbb{P}$ is a positive invariant set for the system, and the system is stable around its equilibrium state.



(a) $\mathbb{P}$ (b) $\boldsymbol{\xi}(\mathbb{P})$

Fig. 12: Positive invariant set of the system $\mathbb{P}$

F. Simulation results

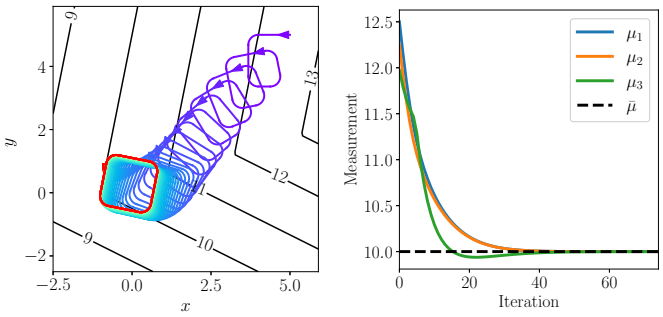
Fig. 13a shows the simulation of the regulated cycle. The cycle is evolving in the plane toward a position where measured depth under the robot converge to the reference depth $\bar{\boldsymbol{\mu}} = \begin{bmatrix}10 & 10 & 10\end{bmatrix}^T$. After a few iterations, the robot is stabilized on its stable cycle, as shown by the measurements evolution in Fig. 13b.

IV. Application

A. Robot description

Cycle control has been tested at the Guerlédan lake on BlueBoats by BlueRobotics, as shown in Fig. 14. BlueBoats are small differential autonomous surface vehicles equipped with navigation sensors: a GNSS receiver, an Inertial Measurement Unit (IMU), and a magnetometer. Exteroceptive sensors are also available, such as an echosounder to measure the depth below the robot.

[3]Note that expression of $\mathbf{A}$ depends on the seafloor. Eigenvalues e_i are therefore determined for the seabed shown in Fig. 10a



(a) Robot's trajectory (b) $\boldsymbol{\mu}$ through iterations

Fig. 13: Simulation of the regulated cycle

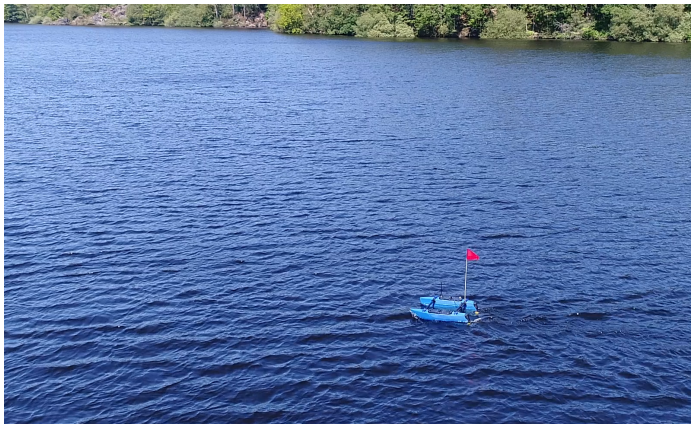


Fig. 14: BlueBoat sailing on Guerlédan lake

In this trial the GNSS receiver is only used to get the ground truth of the robot trajectory. This position is not used in the control loop.

B. Experiment area and cycle description

The Landroanec cove on Guerlédan lake has been chosen as the test area. The seafloor of this cove has been mapped using a multibeam sounder and a digital elevation model has been generated as shown in Fig. 15, but this mapping is not known by the robot.

The BlueBoat is controlled to follow a square cycle in this cove around the reference $\bar{\boldsymbol{\mu}} = \begin{bmatrix}10 & 10 & 10\end{bmatrix}$.

The cycle approach is then applied on the robot, and only three measurements are required to stabilize the cycle on the isobath. Measurements are taken at the same positions than in Fig. 10b to stick to the simulation shown in Fig. 13a.

C. Experimentation results

The BlueBoat is then placed on the lake near the targeted stable area, but not already on the isobath. Two experiments were carried out. The two trajectories of the

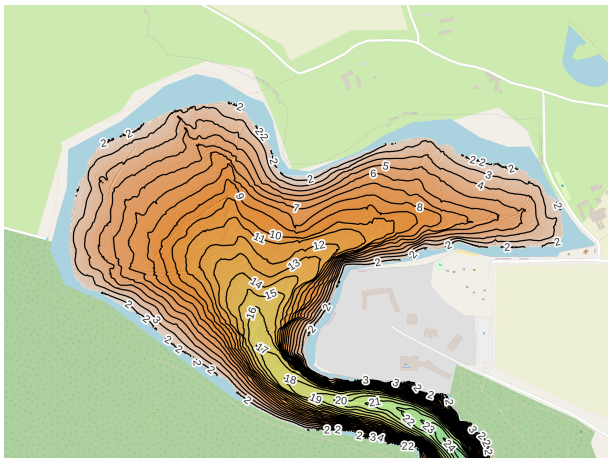
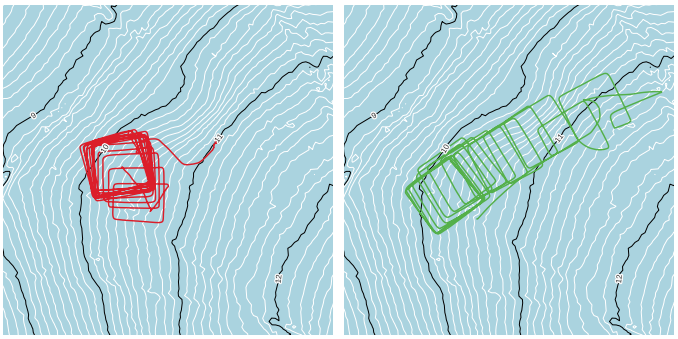


Fig. 15: Digital Elevation Model of the Landroanec cove on Guerlédan lake

BlueBoat for each experiments are shown in Fig. 16a and Fig. 16b.



(a) GNSS Trace 1 (b) GNSS Trace 2

The robot trajectory controlled through cycle iterations is converging toward the reference regardless of its initial pose. The final cycle for each trial seems stable, and the two experiments are converging toward the same area of the lake.

V. Conclusion

Stable cycle is a new paradigm to control dynamical systems with minimal information. Stable cycles allow a robot to navigate in a known environment without getting lost. First, the robot is driven along a cycle. Then this cycle is progressively moved away using measurements in the environment to stabilize it at a pre-defined state. With this method, it is possible to estimate the state of the robot as soon as the cycle is stabilized around its reference.

The limitations of this method may come from the required prior knowledge of the environment. Indeed, to define the stable cycle based on measurements, the map of the environment has to be known. Then, the control of the robot can be implemented, but all initial conditions for the robot do not lead to cycle stabilization either, or could lead to a stable cycle not at the planned position. For now, stability of the method as been proven locally around the equilibrium position and without disturbances.

However, field experiments with the BlueBoat at the Lake of Guerlédan show that the stable cycle paradigm can be used to control a robot to a predefined zone without any external localization system, or in GNSS denied environments. Cycles control could lead to new applications in the field of maritime robotics such as scanning an area by gradually shifting the stable cycle.

References

[1] F. Maurelli, S. Krupiński, X. Xiang, and Y. Petillot, "Auv localisation: A review of passive and active techniques," International Journal of Intelligent Robotics and Applications, vol. 6, no. 2, pp. 246–269, 2022. doi: 10.1007/s41315-021-00215-x.

[2] S. Durand, B. Boisseau, N. Marchand, and J.-F. Guerrero-Castellanos, "Event-Based PID Control: Application to a Mini Quadrotor Helicopter," Journal of Control Engineering and Applied Informatics, vol. 20, no. 1, pp. 36–47, Mar. 2018.

[3] L. Bergantin, N. Harbaoui, T. Raharijaona, and F. Ruffier, "Oscillations make a self-scaled model for honeybees' visual odometer reliable regardless of flight trajectory," Journal of the Royal Society Interface, vol. 18, no. 182, p. 20 210 567, 2021.

[4] K. J. Lohmann and C. M. F. Lohmann, "Orientation and Open-Sea Navigation in Sea Turtles," Journal of Experimental Biology, vol. 199, no. 1, pp. 73–81, Jan. 1996. doi: 10.1242/jeb.199.1.73.

[5] H. K. Khalil, "Nonlinear systems,"

[6] J.-J. E. Slotine and W. Li, "Nonlinear applied control," Li, W., Ed, 1991.

[7] S. H. Strogatz, Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering. CRC press, 2018.

[8] R. Alur and D. L. Dill, "A theory of timed automata," Theoretical computer science, vol. 126, no. 2, pp. 183–235, 1994.

[9] R. Wang et al., "Timed automata based motion planning for a self-assembly robot system," in 2014 IEEE International Conference on Robotics and Automation (ICRA), 2014, pp. 5624–5629. doi: 10.1109/ICRA.2014.6907686.

[10] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," American Journal of Mathematics, vol. 79, no. 3, pp. 497–516, 1957.

[11] J.-P. Aubin, A. Bayen, and P. Saint-Pierre, Viability Theory: New Directions. Jan. 2011. doi: 10.1007/978-3-642-16684-6.

[12] A. Bourgois, A. Chaabouni, A. Rauh, and L. Jaulin, "Proving the stability of the rolling navigation," Acta Cybernetica, vol. 26, no. 1, pp. 5–34, 2023.

[13] L. Jaulin et al., Interval analysis. Springer, 2001.