Navigation without localization using stable cycles

Quentin Brateau*, Fabrice Le Bars†, and Luc Jaulin‡
UMR 6285 Lab-STICC, ROBEX Team
ENSTA, 2 rue François Verny, 29200 Brest
Email: *quentin.brateau@ensta.fr, †fabrice.le_bars@ensta.fr,
‡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, Navigation, Control, Stability

I. Introduction

Classical methods for navigation rely on control laws based on an estimation of the robot position [1]. This position estimate is derived from exteroceptive sensors such as GNSS receivers or acoustic positioning solutions in marine robotics [2], [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 require 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 [3].

Additionally, bio-mimetism is gaining traction in robotics [4], [5], [6]. 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 [7]. 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 [8].

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 positioning 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 [3]. This navigation paradigm could also take advantage of bio-inspired sensors such as the electrical sense [6], or the sky polarization [4].

Using stable cycles for controlling and localizing robotic systems represents a new paradigm. As far as we know, there is no such paradygms available in the literature to navigate in underwater environment without localization. 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 $\mathbf{x} \in \mathbb{R}^n$, an input $\mathbf{u} \in \mathbb{R}^m$, and governed by:

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

where $\mathbf{f}$ is the evolution equation of the system [9], [10]. There exists a flow function [10] $\boldsymbol{\varphi} : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$ which is the solution of (1) with initial condition $\mathbf{x}_0 \in \mathbb{R}^n$, satisfying the following properties:

$$\boldsymbol{\varphi}(\mathbf{x}_0, t_0) = \mathbf{x}(t_0) \tag{2}$$

$$\boldsymbol{\varphi}(\mathbf{x}_0, 0) = \mathbf{x}_0 \tag{3}$$

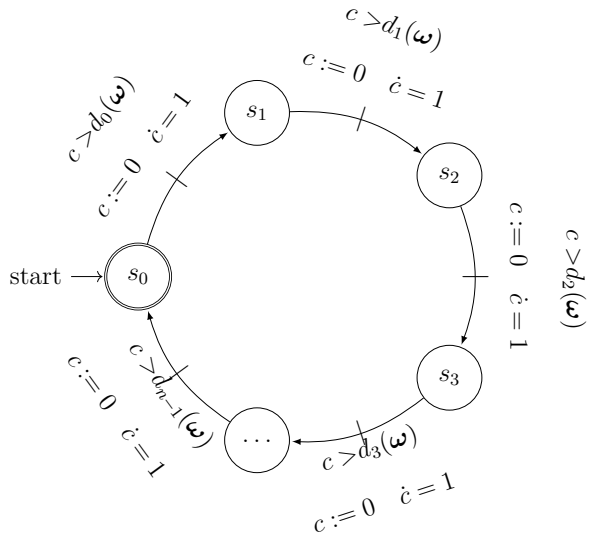
$$\boldsymbol{\varphi}(\boldsymbol{\varphi}(\mathbf{x}_0, t_0), t_1) = \boldsymbol{\varphi}(\mathbf{x}_0, t_0 + t_1). \tag{4}$$

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

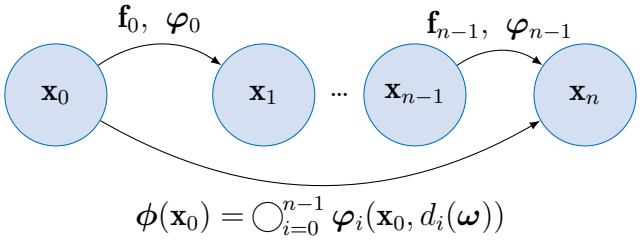
B. Cyclic Timed Automaton

The system can be controlled by a timed automaton as defined in [11], [12]. A timed automaton is a finite state machine extended with a finite collection of real-valued clock variables controlling the transitions between states.

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



(a) Timed automata



(b) Composition of flow functions over a cycle

Fig. 1: Timed automaton and composition of flow function over an iteration

In this automaton, 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 $\mathbf{f}_i$ for each state of the timed automaton, and there exists a flow function $\boldsymbol{\varphi}_i$ 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 automaton. Fig. 1b shows the composition of flow functions over a cycle.

C. Synchronization condition

The duration of one iteration of the timed automaton is denoted by $T(\boldsymbol{\omega})$ and is called the *cyclic period*. The cyclic period is defined by:

$$T(\boldsymbol{\omega}) = \sum_{i=0}^{n-1} d_i(\boldsymbol{\omega}). \tag{5}$$

The dynamical system and the timed automaton are synchronized if after one iteration of the timed automaton the dynamical system come back to the same state. The system therefore satisfies the following condition:

$$\boldsymbol{\phi}(\mathbf{x}(t)) \triangleq \mathbf{x}(t + T(\boldsymbol{\omega})) = \mathbf{x}(t). \tag{6}$$

The block diagram shown in Fig. 2 summarizes the control architecture at this point, where $\mathcal{A}$ represent the automaton and the system is represented by its evolution equation.

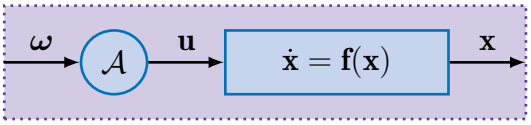


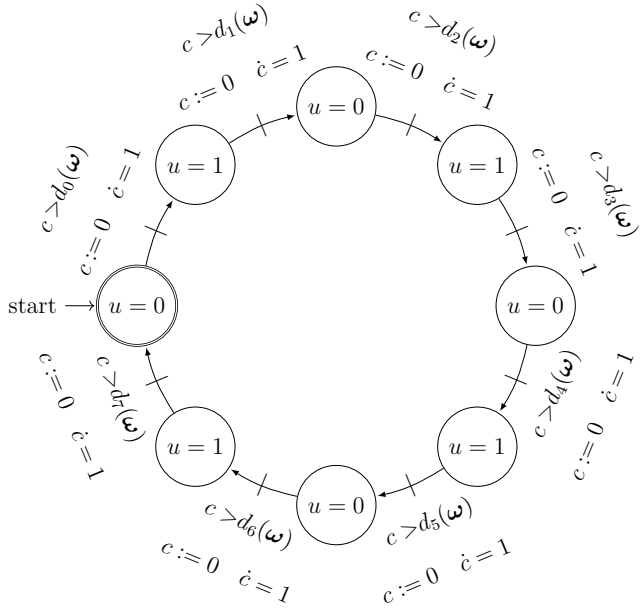
Fig. 2: Block diagram of the robot controlled by a timed automaton

As an example, consider a vehicle following the kinematic model of the unicycle:

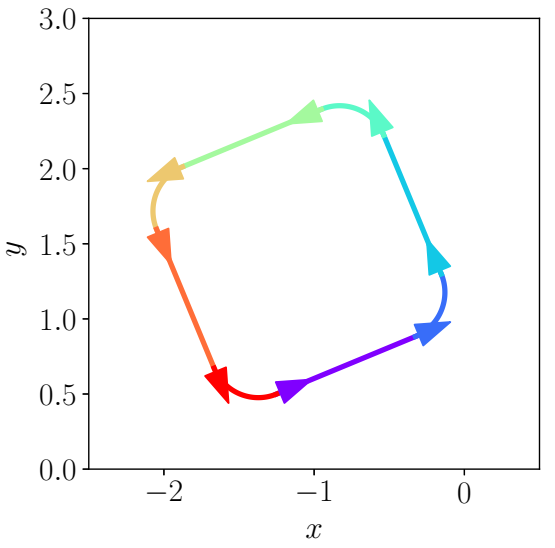
$$\dot{\mathbf{x}} = \begin{cases} \dot{x} = v \cdot \cos(\theta) \\ \dot{y} = v \cdot \sin(\theta) \\ \dot{\theta} = u \end{cases}, \tag{7}$$

where $\mathbf{x} = \begin{bmatrix} x & y & \theta \end{bmatrix}^T$ 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 automaton shown in Fig. 3a is designed to control the trajectory of the vehicle such that the system is following a square cycle as shown in Fig. 3b, by alternating straight lines and circle arcs. As after one iteration of the automaton with a zero input $\boldsymbol{\omega}$ the vehicle comes back to its initial state, the vehicle and the automaton are synchronized.



(a) Square cyclic timed automaton



(b) Square cycle described by the robot trajectory

Fig. 3: Square cycle example

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:

$$\boldsymbol{\eta}_k = \bigcirc_{i=0}^{k-1} \boldsymbol{\phi}(\mathbf{x}, T(\boldsymbol{\omega}_i)). \tag{8}$$

With the cycle abstraction, the block diagram of Fig. 2 is simplified as in Fig. 4.

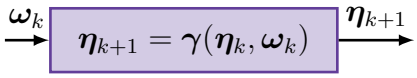


Fig. 4: Block diagram of the controlled cycle

Note that with a non-zero input $\boldsymbol{\omega}_k$, the cycle is destabilized and no longer meets the synchronization condition (6). 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_6, and s_7 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. 5 shows the effect of an input on the cycle state over five iterations. Fig. 5a,Fig. 5b, and Fig. 5c show respectively the effect of inputs $\omega_{k,0}$, $\omega_{k,1}$, and $\omega_{k,2}$, and Fig. 5d 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.

The evolution equation of the square cycle example is modelled by:

$$\boldsymbol{\gamma}(\boldsymbol{\eta}_k, \boldsymbol{\omega}_k) = \boldsymbol{\eta}_k + \begin{bmatrix} -cos(\theta_k) \cdot \omega_{k,0} - sin(\theta_k) \cdot \omega_{k,1} + \frac{-sin(\theta_k) + sin\left(\theta_k + \frac{\pi}{2} \cdot \omega_{k,2}\right)}{\pi} \\ sin(\theta_k) \cdot \omega_{k,0} - cos(\theta_k) \cdot \omega_{k,1} + \frac{cos(\theta_k) - cos\left(\theta_k + \frac{\pi}{2} \cdot \omega_{k,2}\right)}{\pi} \\ \omega_{k,2} \end{bmatrix}. \tag{9}$$

This equation is non-linear and depends on the chosen transition durations adjusted by $\boldsymbol{\omega}_k$. Moreover, inputs $\omega_{k,0}$, and $\omega_{k,1}$ are changing the position of the cycle in the plane relative to the current orientation of the cycle θ_k, while $\omega_{k,2}$ is both changing the position and the orientation of the cycle as shown in Fig. 5c.

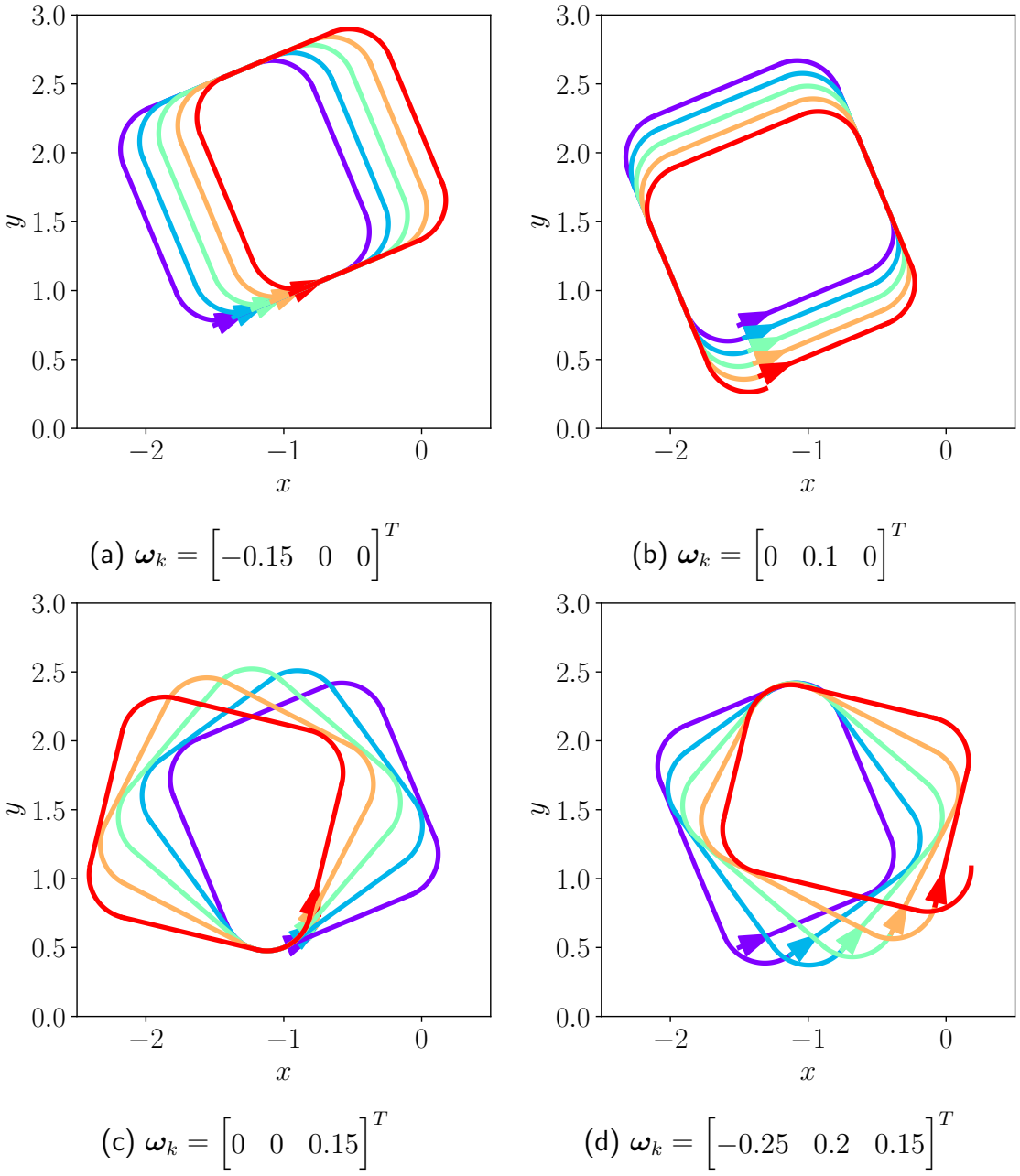


Fig. 5: Cycle control under different inputs

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, $\boldsymbol{\omega}_k$ is computed by:

$$\boldsymbol{\omega}_k = \boldsymbol{\zeta}(\theta_k, \boldsymbol{\nu}_k) = \begin{bmatrix} -cos(\theta_k)\cdot\nu_{k,0} + sin(\theta_k)\cdot\nu_{k,1} - \frac{-sin(\theta_k)+sin(\theta_k+\frac{\pi}{2}\cdot\nu_{k,2})}{\pi} \\ -sin(\theta_k)\cdot\nu_{k,0} - cos(\theta_k)\cdot\nu_{k,1} - \frac{cos(\theta_k)-cos(\theta_k+\frac{\pi}{2}\cdot\nu_{k,2})}{\pi} \\ \nu_{k,2} \end{bmatrix} \tag{10}$$

where θ_k is the orientation of the cycle corresponding to the orientation of the robot at the start of the k^{th} iteration of the robot. It can be measured with a compass or estimated using a state observer for instance. This new stage allows to have a linear relation between the input $\boldsymbol{\nu}_k$ and the output $\boldsymbol{\eta}_k$, which follows:

$$\boldsymbol{\eta}_{k+1} = \boldsymbol{\eta}_k + \boldsymbol{\nu}_k. \tag{11}$$

The block diagram of the cycle controlled in the world frame using this new stage of regulation is shown in Fig. 6.

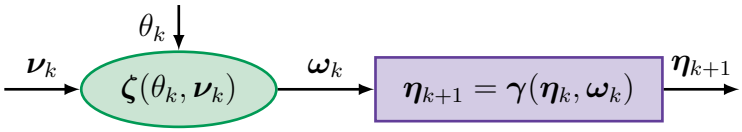


Fig. 6: Block diagram of the controlled cycle

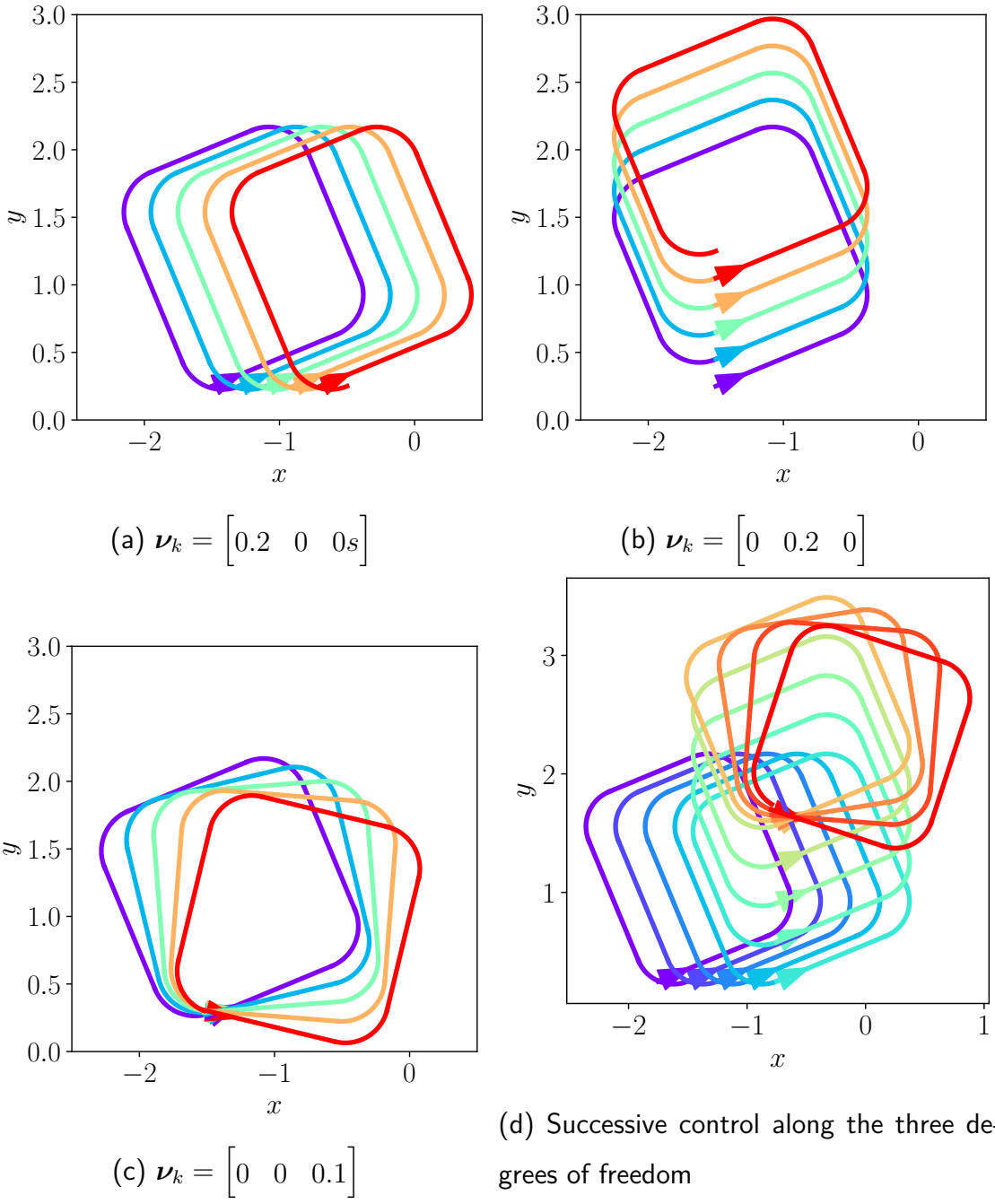
In Fig. 7, a constant input $\boldsymbol{\nu}_{k,0}$, $\boldsymbol{\nu}_{k,1}$, and $\boldsymbol{\nu}_{k,2}$ is applied over 5 iterations on the system and cycles are well moved in the world frame.

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. The system with measurements is then modeled by:

$$\mathcal{S} : \begin{cases} \boldsymbol{\eta}_{k+1} = \boldsymbol{\gamma}(\boldsymbol{\eta}_k, \boldsymbol{\omega}_k) \\ \boldsymbol{\mu}_k = \boldsymbol{\sigma}(\boldsymbol{\eta}_k, \boldsymbol{\omega}_k) \end{cases}. \tag{12}$$

For the square cycle example, the robot is able to sense the depth under itself using an echosounder. The measurement equation used by the robot $g(\mathbf{x})$ is defined by:



(a) $\boldsymbol{\nu}_k = \begin{bmatrix} 0.2 & 0 & 0s \end{bmatrix}$

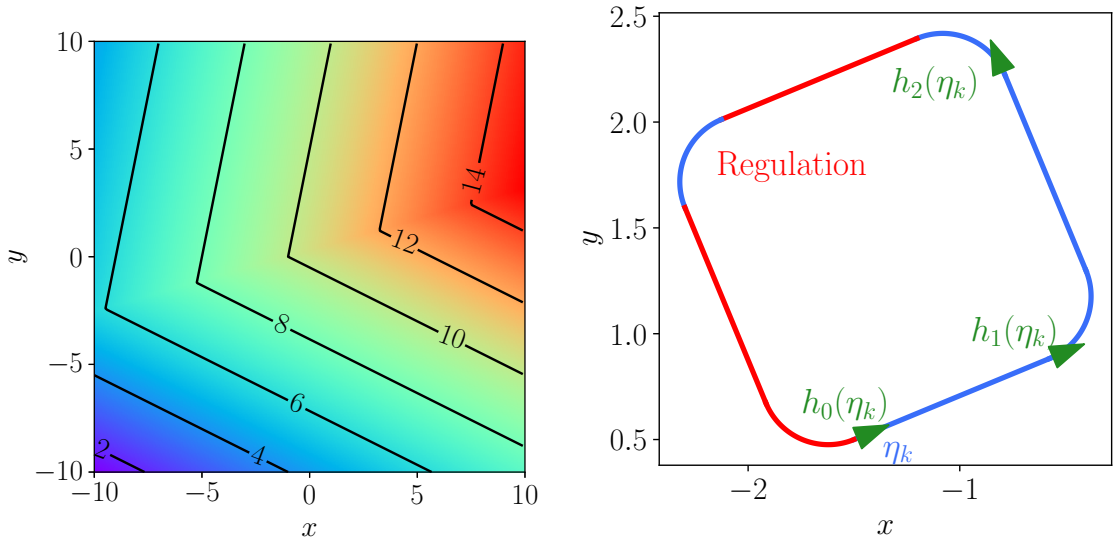
(b) $\boldsymbol{\nu}_k = \begin{bmatrix} 0 & 0.2 & 0 \end{bmatrix}$

(c) $\boldsymbol{\nu}_k = \begin{bmatrix} 0 & 0 & 0.1 \end{bmatrix}$

(d) Successive control along the three degrees of freedom

Fig. 7: Cycle control in the world frame

$$\begin{cases} g(\mathbf{x}) = 10 + \min\limits_{i=0,1} \left\{ 0.1 \cdot \det(\mathbf{b_i} - \mathbf{a_i}, \begin{bmatrix} x & y \end{bmatrix}^T - \mathbf{a_i}) \right\} \\ a_0 = \begin{bmatrix} 0 & 5 \end{bmatrix}^T, \quad b_0 = \begin{bmatrix} -1 & 0 \end{bmatrix}^T \\ a_1 = \begin{bmatrix} -1 & 0 \end{bmatrix}^T, \quad b_1 = \begin{bmatrix} 5 & -3 \end{bmatrix}^T \end{cases} . \tag{13}$$



(a) Simulated seafloor (b) Measurement positions

Fig. 8: Measurements environment and setup

The simulated seafloor is shown in Fig. 8a. Three measurements are taken along the cycle, as shown in Fig. 8b, 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 $\boldsymbol{\varphi}_i$, gives the measurement positions [1]

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. 8a, following:

$$\boldsymbol{\lambda}(\boldsymbol{\mu}_k, \bar{\boldsymbol{\mu}}) = \begin{bmatrix} k_1 & 0 & 0 \\ 0 & 0 & k_2 \\ -k_3 & k_3 & 0 \end{bmatrix} \cdot (\bar{\boldsymbol{\mu}} - \boldsymbol{\mu}_k), \tag{14}$$

[1]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. 8b.

where k_1 modulates the cycle position in the x direction, k_2 in the y direction, and k_3 modulates its orientation. The cycle is now controlled and regulated in the plane to reach the target state $\bar{\boldsymbol{\eta}}$ specified by $\bar{\boldsymbol{\mu}}$. Fig. 9 shows the block diagram of the regulated system.

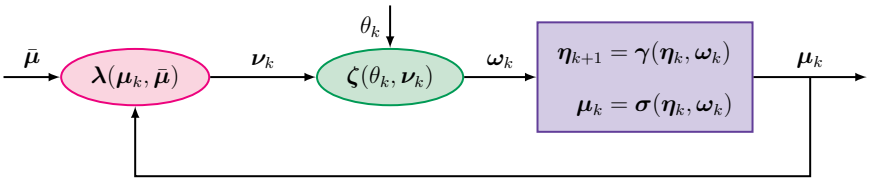


Fig. 9: Block diagram of the autonomous system

From this block diagram is derived the equation of the autonomous system given by:

$$\boldsymbol{\eta}_{k+1} = \boldsymbol{\xi}_{\bar{\boldsymbol{\mu}}}(\boldsymbol{\eta}_k). \tag{15}$$

Equation 15 is linearized around $\bar{\boldsymbol{\eta}}$ such that it can be written as:

$$\boldsymbol{\eta}_{k+1} = \mathbf{A} \cdot (\bar{\boldsymbol{\eta}} - \boldsymbol{\eta}_k). \tag{16}$$

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:

$$\begin{cases} e_0 = 1 - 0.025k_0 - 0.03k_1 + 0.042\sqrt{0.35k_0^2 - k_0k_1 + \dfrac{k_1^2}{2}} \\ e_1 = 1 - 0.025k_0 - 0.03k_1 - 0.042\sqrt{0.35k_0^2 - k_0k_1 + \dfrac{k_1^2}{2}} \\ e_2 = 1 - 0.05 \cdot k_2 \end{cases}. \tag{17}$$

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. [2]

[2]Note that expression of $\mathbf{A}$ depends on the seafloor. Eigenvalues e_i are therefore determined for the seabed shown in Fig. 8a

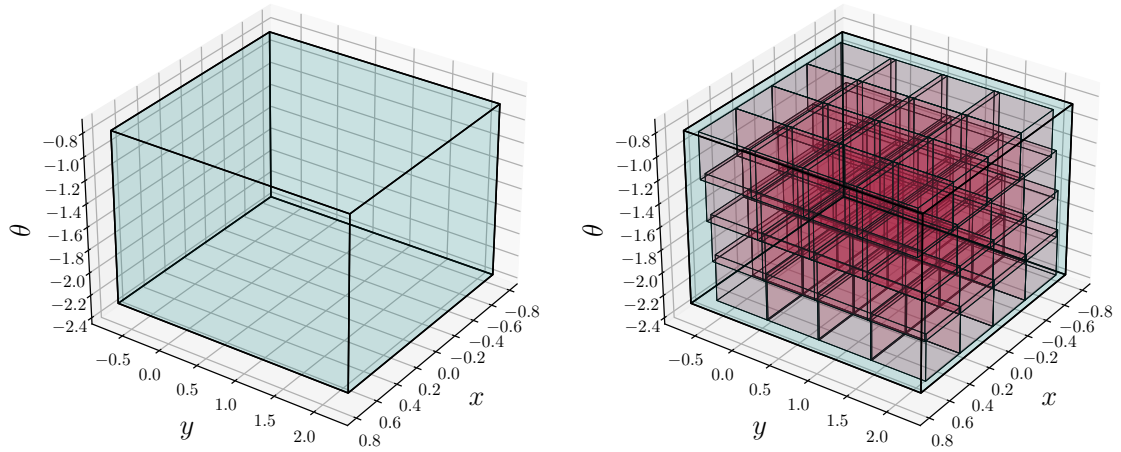
E. Cycle stability

The cycle stability is proven using an interval analysis approach. The proposed approach is to find a positive invariant set of state $\mathbb{P}$ [13], [14] for the autonomous system. This set must satisfy the condition:

$$\boldsymbol{\xi}_{\bar{\boldsymbol{\mu}}}(\mathbb{P}) \subset \mathbb{P}, \tag{18}$$

which means that as soon as the state of the system enters the set $\mathbb{P}$, it is forever captured in it. $\mathbb{P} = \Big[[-0.75, 0.75] \quad [-0.75, 2.25] \quad [-2.32, -0.82]\Big]$ is a positive invariant set for the square cycle example. This box shown in blue in Fig. 10a is split into $n_{split} = 4$ in each dimension to reduce the wrapping effect of interval analysis [15]. The evolution function $\boldsymbol{\xi}_{\bar{\boldsymbol{\mu}}}$ is then applied to each box once, and the resulting boxes are plot in pink in Fig. 10b.

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.



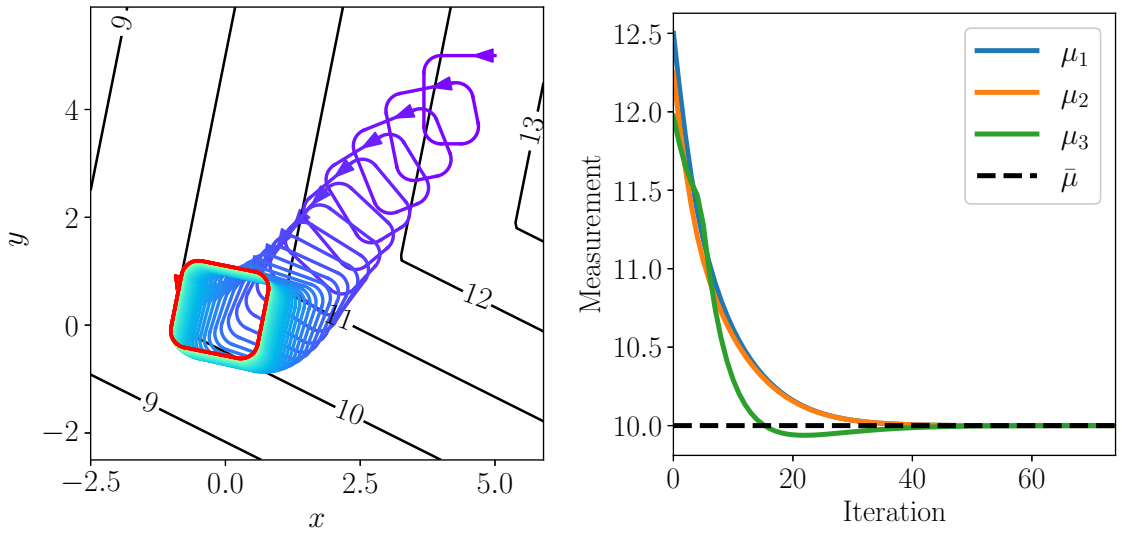
(a) $\mathbb{P}$ (b) $\boldsymbol{\xi}(\mathbb{P})$

Fig. 10: Positive invariant set of the system $\mathbb{P}$

F. Simulation results

Fig. 11a 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. 11b.



(a) Robot's trajectory

(b) $\boldsymbol{\mu}$ through iterations

Fig. 11: Simulation of the regulated cycle

IV. Application

A. Robot description

Cycle control has been tested at the Guerlédan lake on BlueBoats by BlueRobotics, as shown in Fig. 12. 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.

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 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. 13, but this mapping is not known by the robot.

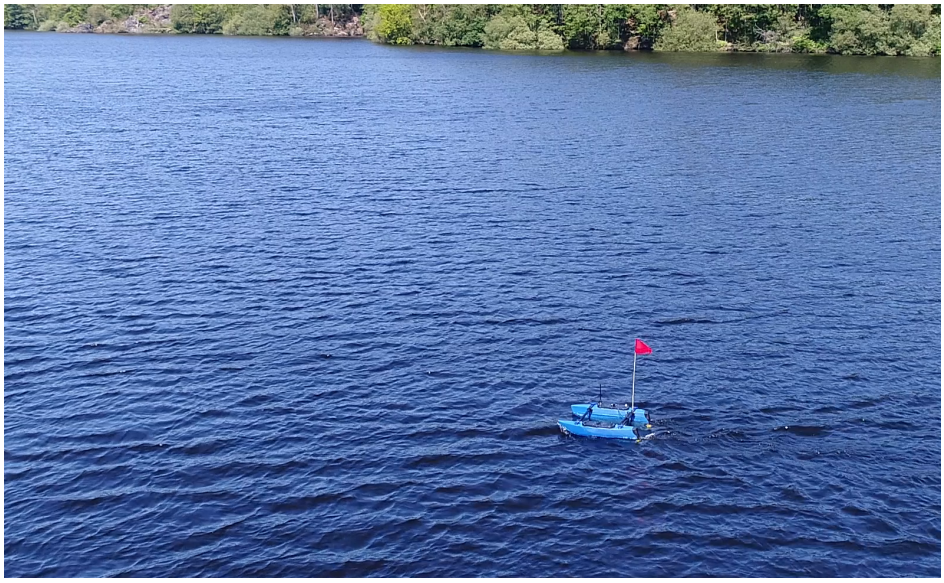


Fig. 12: BlueBoat sailing on Guerlédan lake

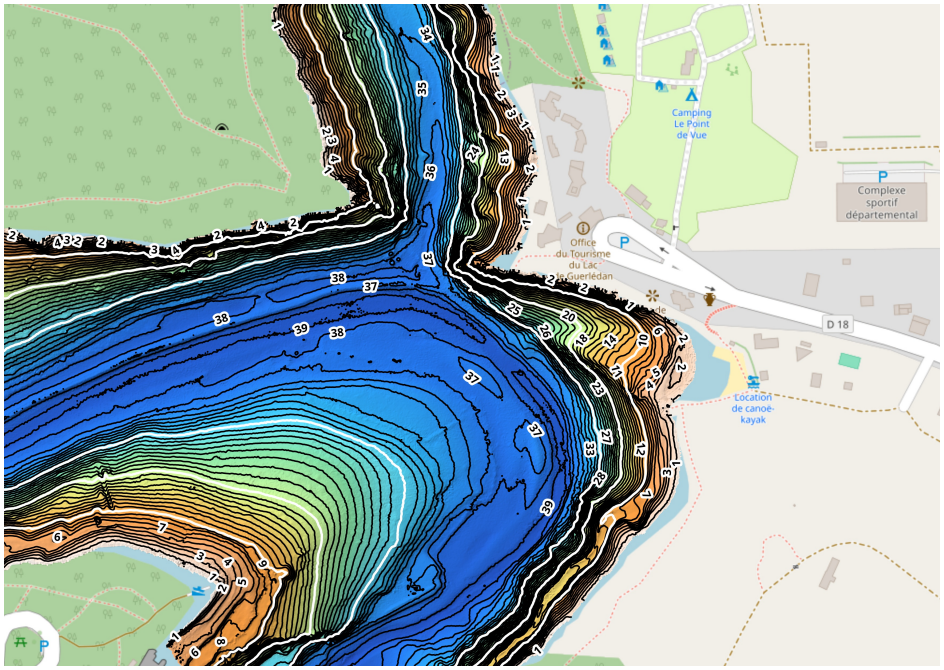


Fig. 13: Digital Elevation Model of the Guerlédan lake

The BlueBoat is controlled to follow a square cycle in this cove around the reference $\bar{\boldsymbol{\mu}} = \begin{bmatrix} 18 & 18 & 18 \end{bmatrix}$.

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 as in Fig. 8b to stick to the simulation shown in Fig. 11a.

C. Experimentation results

The BlueBoat is then placed on the lake near the targeted stable area, but not already on the isobath. Fig. 14 shows the robot trajectory during the experiment. After a few iterations of the timed automaton, the robot approaches the reference depth $\bar{\boldsymbol{\mu}}$. Its trajectory is then stabilized on the chosen isobath.

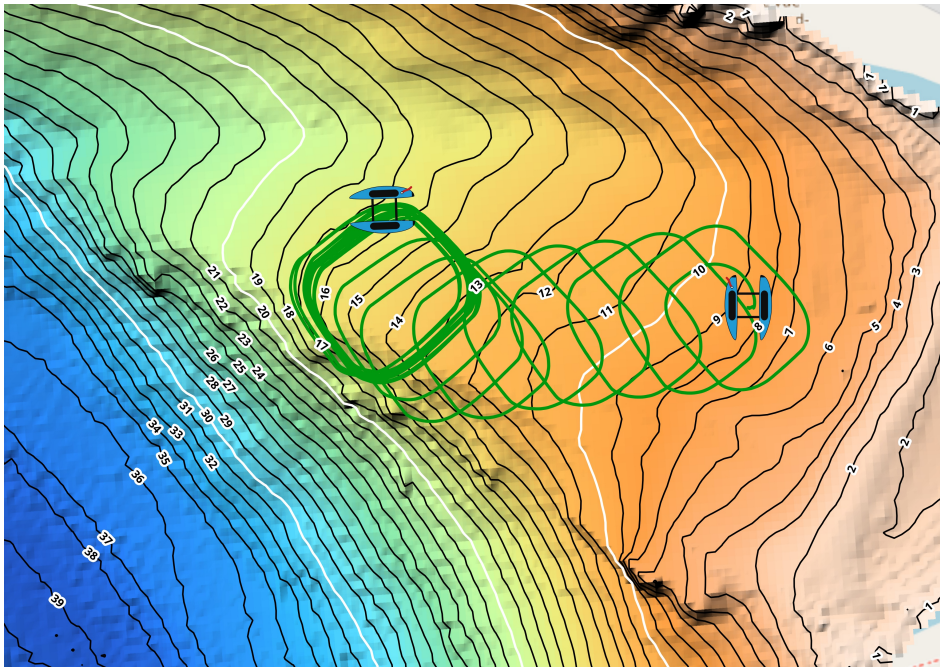


Fig. 14: GNSS Trace 1

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] L. Paull, S. Saeedi, M. Seto, and H. Li, "AUV navigation and localization: A review," *IEEE J. Ocean. Eng*, vol. 39, no. 1, pp. 131–149, Jan. 2014.

[2] F. Maurelli, S. Krupiński, X. Xiang, and Y. Petillot, "AUV localisation: a review of passive and active techniques," *Int. J. Intell. Robot. Appl.*, vol. 6, no. 2, pp. 246–269, Jun. 2022.

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

[4] J. Dupeyroux, S. Viollet, and J. Serres, "An ant-inspired celestial compass applied to autonomous outdoor robot navigation," *Robot. Auton. Syst.*, vol. 117, pp. 40–56, Jul. 2019.

[5] J. Serres and F. Ruffier, "Optic flow-based collision-free strategies: From insects to robots," *Arthropod Struct. Dev.*, vol. 46, no. 5, pp. 703 – 717, Sep. 2017.

[6] F. Boyer, V. Lebastard, S. B. Ferrer, and F. Geffard, "Underwater pre-touch based on artificial electric sense," *Int. J. Robot. Res.*, vol. 39, no. 6, pp. 729–752, Mar. 2020.

[7] L. Bergantin, N. Harbaoui, T. Raharijaona, and F. Ruffier, "Oscillations make a self-scaled model for honeybees' visual odometer reliable regardless of flight trajectory," *J. R. Soc. Interface*, vol. 18, no. 182, p. 20210567, Sep. 2021.

[8] K. J. Lohmann and C. M. F. Lohmann, "Orientation and Open-Sea Navigation in Sea Turtles," *J. Exp. Biol.*, vol. 199, no. 1, pp. 73–81, Jan. 1996.

[9] H. K. Khalil and J. W. Grizzle, "Introduction," in *Nonlinear systems*, 3rd ed. Prentice hall, 2002, ch. 1, pp. 1–34.

[10] S. H. Strogatz, "Linear systems," in *Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering*. CRC press, 2018, ch. 5, pp. 125–145.

[11] R. Alur and D. L. Dill, "A theory of timed automata," *Theor. Comput. Sci.*, vol. 126, no. 2, pp. 183–235, Apr. 1994.

[12] R. Wang, P. Luo, Y. Guan, H. Wei, X. Li, J. Zhang, and X. Song, "Timed automata based motion planning for a self-assembly robot system," in *ICRA*, May 2014, pp. 5624–5629.

[13] A. Benzaouia, "Saturated linear systems: Controller design," in *Saturated Switching Systems*. Springer London, 2012, ch. 2, pp. 37–75.

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

[15] G. Alefeld and G. Mayer, "Interval analysis: theory and applications," *J. Comput. Appl. Math.*, vol. 121, no. 1, pp. 421–464, Sep. 2000.