entropy

MDPI

Article

Novel Bioinspired Approach Based on Chaotic Dynamics for Robot Patrolling Missions with Adversaries

Daniel-Ioan Curiac [1,*], Ovidiu Banias [1], Constantin Volosencu [1] and Christian-Daniel Curiac [2]

[1] Automation and Applied Informatics Department, Politehnica University of Timisoara, 300223 Timisoara, Romania; ovidiu.banias@aut.upt.ro (O.B.); constantin.volosencu@aut.upt.ro (C.V.)

[2] Electrical Engineering and Information Technology Department, Technische Universität München, 80333 Munich, Germany; christian.curiac@tum.de

* Correspondence: daniel.curiac@aut.upt.ro; Tel.: +40-256-403-227

Received: 23 April 2018; Accepted: 16 May 2018; Published: 18 May 2018

Abstract: Living organisms have developed and optimized ingenious defense strategies based on positional entropy. One of the most significant examples in this respect is known as protean behavior, where a prey animal under threat performs unpredictable zig-zag movements in order to confuse, delay or escape the predator. This kind of defensive behavior can inspire efficient strategies for patrolling robots evolving in the presence of adversaries. The main goal of our proposed bioinspired method is to implement the protean behavior by altering the reference path of the robot with sudden and erratic direction changes without endangering the robot's overall mission. By this, a foe intending to target and destroy the mobile robot from a distance has less time for acquiring and retaining the proper sight alignment. The method uses the chaotic dynamics of the 2D Arnold's cat map as a primary source of positional entropy and transfers this feature to every reference path segment using the kinematic relative motion concept. The effectiveness of this novel biologically inspired method is validated through extensive and realistic simulation case studies.

Keywords: positional entropy; protean behavior; mobile robot; adversarial patrolling; chaotic dynamics; Arnold's cat map

1. Introduction

Starting from the classical Shannon's definition of information entropy as a measure of uncertainty in a discrete distribution [1,2], Pal and Pal [3] proposed a new concept that characterizes the localization uncertainty of a particular object in a given scene: *positional entropy*. This type of entropy basically quantifies the navigational unpredictability of an object [4,5] and has higher values in the case of random and unexpected movements [4–6] as the ones that characterize the protean behavior of diverse animal species under threat.

Throughout their evolution over millions of years, living organisms have developed ingenious defense mechanisms to protect themselves against enemies, including camouflage, mimicry, fleeing, hiding, freezing, etc. Of a special interest for robotic path planning in adversarial environments is a particular defense strategy, employing positional entropy, named protean behavior [7–9]. Here, a prey animal under threat pursues an unpredictable zig-zag path, with sudden direction and speed changes meant to confuse, delay and escape the predator. As a consequence, the protean behavior is proved to be effective not only as a response to an ongoing attack, but also as a prevention measure for various animal species including voles and spiny mice [10], cockroaches [11], hares or minnows [12].

When accomplishing patrol missions, mobile robots may face two types of attacks: (a) adversary (e.g., a sniper) may target and destroy the robots from a distance; or, in rare situations; and (b) adversary

may chase and capture the robots. From our perspective, imparting protean behavior to patrol robots can efficiently countermeasure the first type of attacks by offering the enemies less time to acquire and retain the proper sight alignment, while in the case of mobile robots chased by enemies, protean behavior can be envisioned only as a complementary method. In both cases, such strategies may be feasibly implemented within an intelligent path planning mechanism.

Traditionally, the path planning problem in adversarial conditions has been tackled following two different lines of attack: the game theoretic approach; and the path chaotization to ensure higher levels of path unpredictability.

In game theory, the robot motion in the presence of adversaries is basically interpreted as a pursuit-evasion game [13,14] where based on three models (i.e., mathematical models of prey, predator and environment) diverse path planning strategies are implemented. As a result, a series of relevant studies have been reported for different settings involving single-robot [15], multi-robot [16,17], or swarm robotic systems [18] with full or limited information about adversaries [19,20]. An interesting approach in this context is presented in [21] where the protean fleeing is envisioned as a possible solution. However, such game theoretic approaches, even very theoretically challenging and extremely useful for gaining a profound understanding of the real-life scenarios, are still far from being implemented on autonomous robots due to their numerous simplifying assumptions, imprecision and uncertainties associated with the employed mathematical models and computational complexity [22,23].

The second stratagem to cope with opponents is based on the reasonable assumption that an unpredictable path can help securing the robot patrol mission. Derived from the inherent long-term unpredictability provided by chaotic systems due to their sensitivity to the initial conditions characteristics [24–26], these types of approaches cover a large variety of patrolling missions including point of interest surveillance [27,28], perimeter surveillance [29] and area surveillance [30–34]. When speaking about short-term unpredictability that characterizes the protean behavior, the direct use of chaotic dynamics is inefficient but discrete chaotic systems may offer the needed source of path randomness. In this context, in the only reported method to date that tackles the protean behavior [35], we employed the inherent robot's obstacle avoidance mechanism to circumvent chaotically generated Fictive-Temporary Obstacles (FTOs) in the attempt to provide the needed irregular zigzagging path. While the mentioned method utilizes a combined software-hardware mechanism that changes both the path planning algorithm and the robot's obstacle sensing module, in this paper, we tackle the problem from a different angle, the zig-zag movements being generated using only software means (only the path planning algorithm is altered).

In this paper, we provide a biologically inspired method to address the scenarios where the adversaries target the patrol robot from a distance. We start with analyzing the 2D Arnold's cat map (ACM), a simple discrete chaotic system that provides random-like sequences of points. Using the kinematic relative motion concept, the randomness provided by ACM is utilized to change every segment of the reference path into a misleading zig-zag trajectory, thus imparting the required positional entropy by emulating the protean behavior of prey animals under threat. For this, a specially tailored Additional Waypoints Generation (AWG) algorithm is designed. The effectiveness of the AWG-based method is validated in realistic simulation case studies, proving certain advantages over FTO-based method in obstacle-free environments or in environments with small or medium size obstacles. If large obstacles are encountered, the paper suggests a mixture of the two methods.

The rest of the paper is organized as follows. Section 2 presents the problem formulation, giving some preparatory analysis about its future solving. While Section 3 is devoted to the design of additional waypoints generation algorithm, in Section 4, the bioinspired patrolling method is described, offering detailed implementation insights. Section 5 provides two extensive simulation case studies that validate the effectiveness of the proposed method and its advantages over the FTO-based method. In the final section, conclusions and some final remarks are given.

2. Problem Formulation

As proved by various animal species, the protean behavior is an efficient tactic when coping with adversaries [7–9]. Imparting such a behavior to mobile robots accomplishing patrol missions presume sudden direction changes without affecting their long-term assignment. In these conditions, the robot's path planning mechanism will alter its normal trajectory into a misleading path for enemies. By this, the opponents trying to target and destroy the robot from a distance will have less time for acquiring and retaining the proper sight alignment. Our strategy is based on inserting supplementary waypoints to obtain small zig-zag movements that emulate the behavior of individual prey animal under threat.

Problem statement: *Let us consider an autonomous mobile robot that accomplishes a given patrolling task in a two-dimensional workspace* $W \subset \mathbb{R}^2$ *containing a set of n fixed obstacles* O_i, $i = 1, \ldots, n$. *The robot is outfitted with onboard sensing and computation capabilities. The assignment is to adjust the robot's path using only onboard capabilities in order to cope with adversaries, without jeopardizing its basic mission.*

In [35], a similar problem is solved by employing a combined hardware-software method that simulates sequences of fictive and temporary obstacles. In this paper, we target a pure software solution for this problem that will change only the path planning algorithm. Our idea is to utilize the chaotic dynamics of 2D Arnold's Cat Map to generate additional waypoints without significantly modifying the overall shape of the trajectory. By this, we ensure the required short-term unpredictability of the robot path in adversarial conditions, similar to the protean behavior of some species of prey animals when being chased by enemies.

The proposed approach may be straightforwardly implemented on mobile robots evolving in unknown environments and endorsed with suitable obstacle avoidance mechanisms (e.g., bug-type algorithms like Bug0, OneBug, Distbug, LeaveBug and TangentBug [36]). If the autonomous robots are equipped with enough onboard computational power, they may handle the processing of temporary maps, and, by this, the proposed technique may be implemented even in combination with global path planning algorithms like Breadth-first search, Depth-first search, A*, Dijkstra's algorithm, or Potential Field [37].

3. Additional Waypoints Generation Algorithm

3.1. Preliminaries

Let $P_i(x_i, y_i)$ and $P_{i+1}(x_{i+1}, y_{i+1})$ denote two successive waypoints of the robot trajectory. Our task is to replace the optimal line segment with a zig-zag path to impart short-term unpredictability to robot motion. This protean behavior is produced by a sequence of M_i newly generated supplementary waypoints that are included in the robot path from P_i to P_{i+1}. We will denote these supplementary waypoints with Q_{im}, with $m = 1, 2, \ldots, M_i$.

In general, two different ways can be pursued to obtain the sequence of random-looking waypoints: to use a stochastic approach based on a random number generator to produce a kind of random-walk; or to use a pure deterministic approach employing a chaotic system. When involved in a patrol mission, the location of the autonomous robot must be unpredictable for opponent entities but known to ally entities (e.g., mission coordinators or other robots from the same team). In the case of a deterministic approach ally entities, having complete information about the robot's path planning mechanism, may recalculate the robot trajectory and later use this knowledge for decision-making purposes. Based on this requirement, our proposed AWG algorithm will provide the sequence of waypoints in a deterministic fashion using Arnold's cat map, a 2D chaotic system that offers simplicity in terms of required computations and a random-like distribution of waypoints within the unit square [38].

3.2. Arnold's Cat Map

The additional waypoints will be generated such as the robot trajectory inside a specified zone to appear random and unpredictable for adversaries or external observers. For this, we utilized the inherent unpredictability provided by any chaotic system due to its sensitivity to the initial conditions property [24–26]. An effective solution in this respect is to employ the discrete chaotic dynamics provided by the hyperbolic toral automorphism, known as Arnold's cat map [39,40]. This discrete 2D map is one of the most investigated examples of dynamical systems mainly due to its broad range of applications in image encryption and watermarking [41]. It is described by the following equations:

$$\begin{cases} \hat{x}_j = (\hat{x}_{j-1} + \hat{y}_{j-1}) mod 1 \\ \hat{y}_j = (\hat{x}_{j-1} + 2 \cdot \hat{y}_{j-1}) mod 1 \end{cases} \quad (1)$$

Here, the $mod1$ operator (modulo-1) limits the $\hat{x}_j$ and $\hat{y}_j$ state variables to the interval [0, 1). On $\mathbb{R}^2 \backslash \mathbb{Z}^2$, the transformation (1) describes an obvious chaotic behavior [42] shown by its maximal Lyapunov exponent $\lambda_1 = ln\left(0.5 \cdot \left(3 + \sqrt{5}\right)\right) = 0.9624 > 0$, while the other Lyapunov exponent is $\lambda_2 = ln\left(0.5 \cdot \left(3 - \sqrt{5}\right)\right) = -0.9624 < 0$. Another important characteristic of this map is that the determinant of its linear part is equal to 1, demonstrating that the transformation is area-preserving and, of course, invertible [42].

3.3. Emulating Zig-Zag Movements

The starting point of our approach is the graphical representation of Arnold's cat map (Figure 1), a discrete chaotic system that generates a sequence of random-looking points inside the unit square.

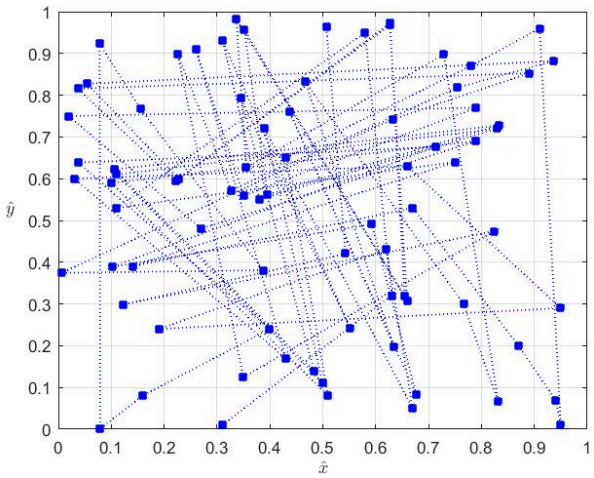

Figure 1. Arnold's cat map.

In order to emulate the protean behavior, we will employ a simple kinematic concept—relative motion [43]. This concept is used to translate the representation of a motion developed inside a mobile coordinate system into coordinates of a fixed coordinate system. Our stratagem can be described by the following situation: we consider two Carthesian coordinate systems: (a) one fixed; and (b) one mobile in which the ACM evolves. The two coordinate systems have collinear horizontal axes, and the mobile one, where the ACM evolves, is dragged with a constant speed $\vec{v}$ along the horizontal axis of the fixed coordinate system (Figure 2).

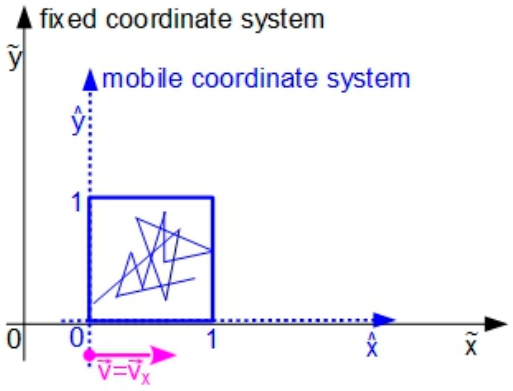

Figure 2. Obtaining the compound motion.

Such a compound motion will be described by the following set of equations:

$$\begin{cases} \hat{x}_j = (\hat{x}_{j-1} + \hat{y}_{j-1}) mod 1 \\ \hat{y}_j = (\hat{x}_{j-1} + 2 \cdot \hat{y}_{j-1}) mod 1 \\ \widetilde{x}_j = \hat{x}_j + v \cdot j \\ \widetilde{y}_j = \hat{y}_j \end{cases} \quad (2)$$

where the first two equations represent the trajectory described by ACM in the mobile system $(\hat{x}, \hat{y})$, while the other two equations describe the coordinate translation from mobile to the fixed coordinate system $(\widetilde{x}, \widetilde{y})$.

By this, a compound motion that emulates the zig-zag shape, which characterizes a protean behavior is obtained. To have a more realistic perspective about this transformation of ACM, we plotted the obtained trajectory for $v = 0$, $v = 0.25$ and $v = 0.5$ in Figure 3.

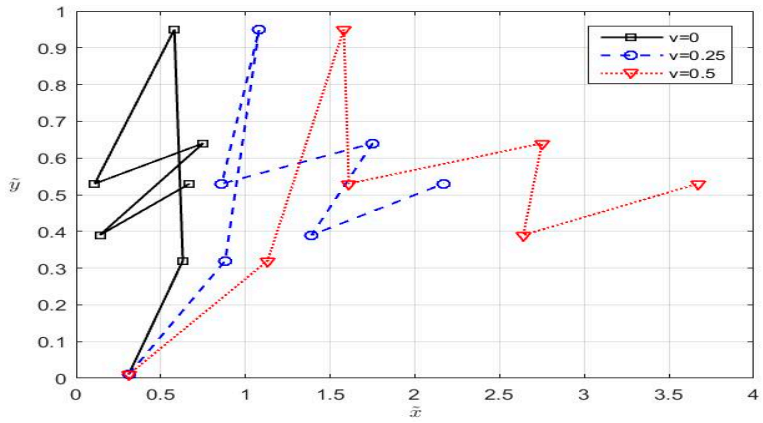

Figure 3. The compound motion inside the fixed coordinate systems for different values of v.

3.4. Adapting the Zig-Zag to a General Segment of Robot Trajectory

By visually analyzing Figure 3 and knowing the feature of ACM to provide random-like points inside the unit square [38], we can state that the strategy presented in the last subsection can provide random waypoints around the $\widetilde{y} = 0.5$ line, starting with $(0,\ 0.5)$. If we want to provide additional waypoints around a general P_iP_{i+1} segment, the system (2) must be adapted using a compound affine transform Λ, while the vector $\vec{v}_i$ will have the same direction as the vector $\vec{P_iP_{i+1}}$ with components along both axes, as depicted in Figure 4. These required changes will affect only the last two equations of (2).

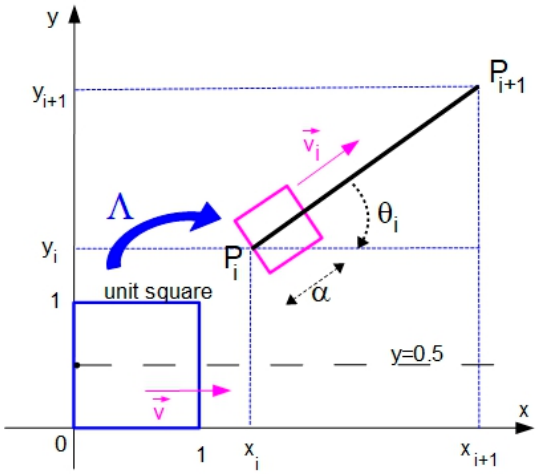

Figure 4. Adapting the motion to a general segment P_iP_{i+1}.

The affine transform Λ used to modify the coordinates provided by (1) is composed of a translation, a scaling and a rotation and has the following transformation matrix (in homogeneous coordinates):

$$T_\Lambda = T * S * R = \begin{bmatrix} 1 & 0 & x_i + 0.5{\cdot}\alpha{\cdot}\sin(\vartheta_i) \\ 0 & 1 & y_i - 0.5{\cdot}\alpha{\cdot}\cos(\vartheta_i) \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha & 0 & 0 \\ 0 & \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(-\vartheta_i) & \sin(-\vartheta_i) & 0 \\ -\sin(-\vartheta_i) & \cos(-\vartheta_i) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (3)$$

where $\vartheta_i = \text{atan}\left(\frac{y_{i+1}-y_i}{x_{i+1}-x_i}\right)$ and α is a constant that characterizes the area where the points will be spread.

Using (3), the transformation Λ of a point generated by ACM inside unit square and having the coordinates $(\hat{x}_j, \hat{y}_j)$ into a point inside a general square (Figure 4) can be written as follows:

$$\begin{bmatrix} x_j \\ y_j \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha{\cdot}\cos(-\vartheta_i) & \alpha{\cdot}\sin(-\vartheta_i) & x_i + 0.5{\cdot}\alpha{\cdot}\sin(\vartheta_i) \\ -\alpha{\cdot}\sin(-\vartheta_i) & \alpha{\cdot}\cos(-\vartheta_i) & y_i - 0.5{\cdot}\alpha{\cdot}\cos(\vartheta_i) \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{x}_j \\ \hat{y}_j \\ 1 \end{bmatrix} \quad (4)$$

or

$$\begin{cases} x_j = \alpha{\cdot}\cos(-\vartheta_i){\cdot}\hat{x}_j + \alpha{\cdot}\sin(-\vartheta_i){\cdot}\hat{y}_j + x_i + 0.5{\cdot}\alpha{\cdot}\sin(\vartheta_i) \\ y_j = -\alpha{\cdot}\sin(-\vartheta_i){\cdot}\hat{x}_j + \alpha{\cdot}\cos(-\vartheta_i){\cdot}\hat{y}_j + y_i - 0.5{\cdot}\alpha{\cdot}\cos(\vartheta_i) \end{cases} \quad (5)$$

In order to obtain the adapted versions of the last two equations from (2), we have to integrate in (5) the effect produced by dragging the mobile system with the constant speed vector $\vec{v}_i$ that has the same direction and orientation as the vector $\vec{P_iP_{i+1}}$. By this, we obtain:

$$\begin{cases} x_j = \alpha{\cdot}\cos(-\vartheta_i){\cdot}\hat{x}_j + \alpha{\cdot}\sin(-\vartheta_i){\cdot}\hat{y}_j + x_i + 0.5{\cdot}\alpha{\cdot}\sin(\vartheta_i) + v_{i,x}{\cdot}j \\ y_j = -\alpha{\cdot}\sin(-\vartheta_i){\cdot}\hat{x}_j + \alpha{\cdot}\cos(-\vartheta_i){\cdot}\hat{y}_j + y_i - 0.5{\cdot}\alpha{\cdot}\cos(\vartheta_i) + v_{i,y}{\cdot}j \end{cases} \quad (6)$$

where $v_{i,x}$ and $v_{i,y}$ are the projections of the speed vector $\vec{v}_i$ (associated with the vector $\vec{P_iP_{i+1}}$) on the axes, which can be either positive or negative. By replacing the last two equations from (2) with equations provided by (6), we will obtain the final form of the system used to generate M_i additional waypoints:

$$\begin{cases} \hat{x}_j = (\hat{x}_{j-1} + \hat{y}_{j-1}) mod 1 \\ \hat{y}_j = (\hat{x}_{j-1} + 2 \cdot \hat{y}_{j-1}) mod 1 \\ x_j = \alpha \cdot \cos(-\vartheta) \cdot \hat{x}_j + \alpha \cdot \sin(-\vartheta) \cdot \hat{y}_j + x_i + 0.5 \cdot \alpha \cdot \sin(\vartheta) + v_{i,x} \cdot j \\ x_j = -\alpha \cdot \sin(-\vartheta) \cdot \hat{x}_j + \alpha \cdot \cos(-\vartheta) \cdot \hat{y}_j + y_i - 0.5 \cdot \alpha \cdot \cos(\vartheta) + v_{i,y} \cdot j \end{cases} \quad with\ j = 1 \ldots M_i \tag{7}$$

Figure 5 presents the path adaptation to the P_iP_{i+1} segment, where $P_i(1, 4)$ and $P_{i+1}(5, 1)$, when $\alpha = 2$ and the magnitude of speed vector $v = 0.5$:

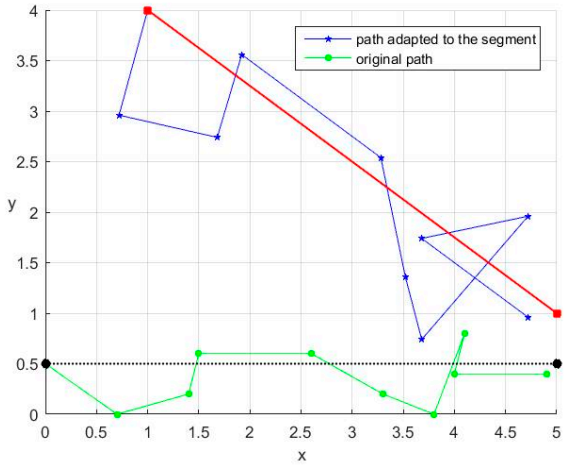

Figure 5. Adapting the path to P_iP_{i+1} segment.

For a given segment P_iP_{i+1} of the reference path, the initialization of system (7) includes the selection of the following parameters: initialization values for ACM $(\hat{x}_0, \hat{y}_0)$; the number M_i of additional waypoints generated for the segment P_iP_{i+1}; the constant α that characterizes the spreading of the additional waypoints around the segment P_iP_{i+1} (each supplementary waypoint will be generated within an $\alpha/2$ distance from the unaltered path as described by Figure 5); and the magnitude of speed vector $\vec{v}_i$. Details about the selection of these values will be given in Section 4.

3.5. Pseudocode of AWG Algorithm

Based on (7), we can provide the additional waypoints that will alter the segment between two successive waypoints P_i and P_{i+1} of initial trajectory using the following pseudocode for the AWG algorithm:

```
1. [Q_{i,1}, ... Q_{i,M}, x̂_{M_i}, ŷ_{M_i}] = AWG (P_i, P_{i+1}, M_i, x̂_0, ŷ_0, v, α,)
2. {
3.    ϑ_i = atan((y_{i+1} − y_i)/(x_{i+1} − x_i));
4.    v_{i,x} = sgn(x_{i+1} − x_i)·|v·cos(ϑ_i)|;
5.    v_{i,y} = sgn(y_{i+1} − y_i)·|v·sin(ϑ_i)|;
6.    for j = 1 to M_i
7.       iterate system (7);
8.    return Q_{i,1}, ... Q_{i,M}, x̂_{M_i}, ŷ_{M_i};
9. }
```

The function AWG receives seven parameters (the coordinates of the two successive waypoints P_i and P_{i+1}; the number M_i of additional waypoints that must be generated for P_iP_{i+1} segment; the initialization values for ACM $(\hat{x}_0, \hat{y}_0)$; the magnitude of speed v; and the value α that characterizes the spreading of the additional waypoints) and provides the sequence of additional waypoints for P_iP_{i+1} segment ($Q_{i,j}$ with $j = 1, \ldots, M_i$) and the last coordinates inside the unit square obtained by ACM (to be used as initial values for ACM generator for the following segment). The pseudocode starts by

calculating the angle ϑ_i and the components of vector $\vec{v}_i$ along axes ($v_{i,x}$, $v_{i,y}$). After that, system (7) is iterated to provide the coordinates of the additional waypoints $Q_{i,1}, \ldots, Q_{i,Mi}$ and the values for the coordinates of the last point offered by ACM ($\hat{x}_M$, $\hat{y}_M$)—these values will be used to initialize the ACM (first two equations of (7)) when computing the additional waypoints for the following segment ($P_{i+1}P_{i+2}$).

4. Patrolling Method Implementation

Let us consider a general case where the mobile robot is tasked with a patrol mission described by a sequence of waypoints P_i, with $i = 1, \ldots, N$, and the corresponding maximum time t_{max} to accomplish this mission. Based on the AWG function presented above, we can split the implementation of the bioinspired robot motion mechanism in three stages: initialization, offline computations and mission accomplishment.

Phase 1: Initialization

This stage includes the selection of the following parameters:

- set of fixed waypoints along the unaltered route (P_i, $i = 1, \ldots, N$) and the maximum time to accomplish the mission (t_{max}); These parameters are the basic descriptors of the robot's patrolling mission and their selection is closely linked with mission's objectives and constraints.
- initial values for ACM $(\hat{x}_0, \hat{y}_0)$; As mentioned in Section 3.2, we must choose $(\hat{x}_0, \hat{y}_0) \in \mathbb{R}^2 \setminus \mathbb{Z}^2$ to obtain a chaotic behavior, an appropriate selection being $(\hat{x}_0, \hat{y}_0) \in (0,1)x(0,1)$;
- parameter α; In order to select the value α that characterizes the spreading of the additional waypoints, two aspects must be considered: (a) a higher value for α will provide a higher unpredictability to the robot path; and (b) a higher value for α will need a higher robot speed to accomplish the mission in the required time (t_{max}). Knowing the robot's speed v_{robot} (has to be as high as possible to increase the effectiveness of the method), we can compute an estimate of the maximum value for α (denoted with α_{max}) by simulating a sufficiently long altered path (our method provides a random-looking sequence of waypoints, which basically depends on initialization values for ACM, so the covered distance may vary) to obtain a better approximation. Having the estimate α_{max} we can select a value using

$$\alpha < \alpha_{max} \tag{8}$$

  a practical example being given in Section 5.

The parameters set in this phase can be viewed as components of a secret-key that enables the ally entities (other robots from the same team or team's coordinators) to have an accurate prediction of the robot's location and its future movements.

Phase 2: Offline preliminary calculations

Before starting the patrol mission, we can derive two of the required parameters to initialize the robot path planning algorithm, namely the magnitude of speed v; and the number M_i of additional waypoints that must be generated for each segment P_iP_{i+1} of the trajectory:

- the magnitude of the speed v (average speed to cover the unaltered path) can be computed using:

$$v = \frac{d}{t_{max}} = \frac{\sum_{i=1}^{N-1} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}}{t_{max}} \tag{9}$$

  where d is the distance covered by the robot on the reference (unaltered) path and can be calculated by summing the length of all segments P_iP_{i+1} with $i = 1, \ldots, N-1$;

- The number M_i of additional waypoints is influenced by the length of the segment P_iP_{i+1} and the magnitude of the speed v used to traverse this segment.

$$M_i = \left\lfloor \frac{\sqrt{(x_{i+1}-x_i)^2+(y_{i+1}-y_i)^2}}{v \cdot \tau} - 1 \right\rfloor = \left\lfloor \frac{\sqrt{(x_{i+1}-x_i)^2+(y_{i+1}-y_i)^2}}{v} - 1 \right\rfloor \quad (10)$$

where the time τ to cover each of the M_i subdivision of the segment P_iP_{i+1} is equal to one time sample ($\tau = 1$), while the notation $\lfloor \rfloor$ denotes the floor function, also known as greatest integer function.

Phase 3: Mission accomplishment and online calculations

In this stage, the robot is autonomously moving from one waypoint to another to accomplish its given missions. The additional waypoints, generated using the AWG algorithm, are inserted between original waypoints to obtain a deceptive path that emulates the protean behavior of prey animals when chased by predators. This stage normally includes the additional waypoints calculations, but if the original trajectory is short enough, these calculations can be done offline in the second phase.

The bioinspired robot motion can be implemented using the following pseudocode:

```
1.  initialize {P_i, i = 1, ... , N }, x̂_0, ŷ_0, α, t_max
2.  compute v;          // Equation (9) is used
3.  for (i = 1; i < N; i++)
4.     compute M_i; //compute the number of additional waypoints for P_iP_{i+1} segment using Equation (10)
5.  start moving;
6.  for (i = 1; i < N; i++){
7.  {
8.     [Q_{i,1}, ... Q_{i,M}, x̂_{Mi}, ŷ_{M_i}] = AWG(P_i, P_{i+1}, M_i, x̂_0, ŷ_0, v, α,); // generates additional the waypoints
9.     for (k = 1; k <= M_i; k++)
10.       Move_towards_waypoint(Q_{i,k}); //the robot is heading towards the additional waypoint
11.    Move_towards_waypoint(P_{i+1});      //the robot is heading towards the end of the segment
12.    x̂_0 = x̂_{Mi};   //prepare the initialization parameters for the following line segment
13.    ŷ_0 = ŷ_{Mi};
14. }
```

In this pseudocode, the Move towards waypoint (P) function represents a motion planning and obstacle avoidance algorithm used to reach the destination P. In this regard, two types of approaches may be considered: global path planning and local path planning. Global path planning algorithms, such as Breadth-first search, Depth-first search, A*, Dijkstra's algorithm, or Potential Field [37], consider completely known maps of the environment. On the other hand, the local path planning schemes that include bug-like algorithms [36,44,45], visibility binary tree algorithm [46] or methods based on perception-action map learning [47], use current local environment information and are designed to tackle the scenarios where the map of environment is unknown or dynamic. For exemplifying reasons, we will use an effective and generally applicable bug-like obstacle avoidance algorithm for unknown environments, named TangentBug [48]. Similar to other algorithms from the bug family [36,49], TangentBug provides paths made up of two motion types: motion-to-goal and boundary-following motion. In the case that no obstacle is encountered, the robot follows a straight line until it reaches the goal (motion-to-goal). In the case that an obstacle is encountered, the robot circumnavigates the obstacle until it finds a direct straight line to the goal (boundary-following).

In addition, the Move towards waypoint (P) function includes a procedure to drop the additional waypoints that fall inside obstacles either by directly verifying their position in case the robot knows the exact position, shape and size of the obstacles or by setting a time limit to reach them (e.g., two times higher than the time to reach it in an environment without obstacles) in case of unknown obstacles.

5. Simulation Results

For evaluating the effectiveness and applicability of our method, two extensive Matlab (version R2016b, MathWorks Inc., Natick, MA, USA) simulation case studies have been carried out. The first one exemplifies the proposed method in all its details, while the second one compares the method with the one based on fictive temporary obstacles generation provided by [35].

5.1. First Case Study

We considered a square workspace (18 m $\times$ 18 m), where we defined a basic octagonal patrolling contour described by the following nine waypoints placed in the vertices: $P_1(5, 0)$, $P_2(0, 5)$, $P_3(0, 10)$, $P_4(5, 15)$, $P_5(10, 15)$, $P_6(15, 10)$, $P_7(15, 5)$, $P_8(10, 0)$ and $P_9(5, 0)$. The mobile robot is equipped with an obstacle detection sensor having a range of 0.5 m and is moving with a maximum velocity of 1.0 m/s. The robot is tasked to cover this closed contour in no more than t_{max} = 120 s.

Knowing the initial set of waypoints P_i and the time t_{max}, in the initialization phase, we need to set the initial values for ACM and to select a suitable value for α. The initial values for ACM are set as $(\hat{x}_0, \hat{y}_0) = (0.4,\ 0.644)$. For obtaining a suitable value α, in Figure 6, we represented the speed needed to cover the trajectories for different values of α. Because, in our simulation scenario, the robot's maximum speed is 1.0 m/s we can choose $\alpha = 1.7$. It is worth mentioning that, for increasing the effectiveness of the method, α and also v_{robot} must take values as high as possible while respecting the limitations imposed by terrain and robot's remaining energy. Using these initialization parameters, we can follow the pseudocode to obtain the altered path presented in Figure 7, where the needed velocity for the robot is 0.9651.

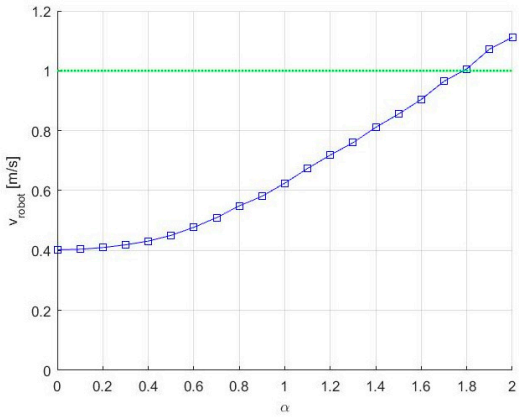

Figure 6. Selecting a suitable value for α.

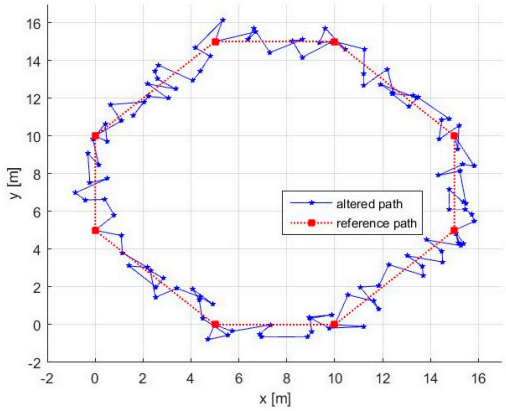

Figure 7. Altered path and reference path.

Remarks:

(a) As presented in Figure 7, in many circumstances, the altered path has a "go-backward" component. This can be observed anytime $v < \alpha$, this inequality being a direct result of the way AWG was designed. In our case, using Equation (9), we obtain $v = 0.4024$, so the mentioned inequality is satisfied.

(b) Theoretically, the path always touches the initial waypoints. In practice, the robot has an allowed position error range, which means that the waypoints are considered as reached when the robot is within a circle around the waypoint with a radius smaller than the allowed position error.

(c) The method can simply deal with missions where only some parts of the path are under threat. In this case, we need: (i) to set initial waypoints anytime the robot enters or leaves the danger zone; and (ii) to generate additional waypoints using our method only for the segments that lay inside the danger zone (Figure 8).

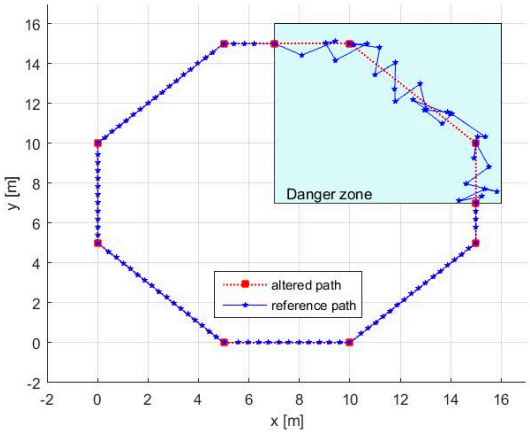

Figure 8. The path in danger and safety zones.

To demonstrate the unpredictability of the path, in Figure 9, three consecutive laps are presented for the same octagonal contour, the use of ACM guaranteeing that the path cannot be periodic.

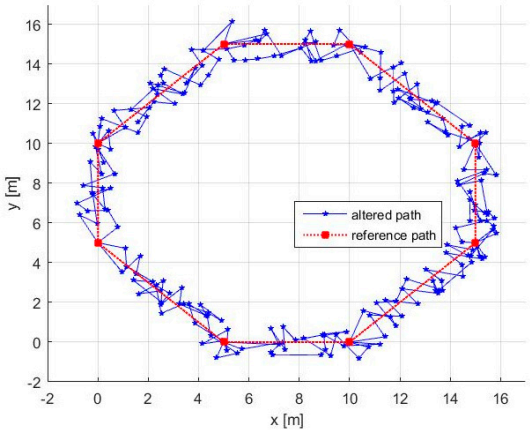

Figure 9. Three laps of the path.

In order to numerically illustrate the unpredictability of the path, we used an ensemble of four metrics, which offers a holistic view about the effectiveness of the proposed method: mean absolute error (MAE), average number of direction changes (AMDC) in a given time interval, the mean absolute angle (MAA) for direction changes and the range of angles (RA) for direction changes.

An appropriate metric that can be used to evaluate the overall unpredictability of the robot path is the Mean Absolute Error (MAE) between two sequences of points: (a) the L equally distanced points P_l^* with $l = 1, \ldots, L$ placed on the reference path C^*; and (b) the corresponding L time-equidistant points P_l' placed on the altered path C'. This metric reflects the robot trajectory deviation from its reference path and is computed by the following formula:

$$MAE(C^*, C') = \frac{1}{L} \sum_{l=1}^{L} d(P_l^*, P_l') \tag{11}$$

where $d(P_l^*, P_l')$ is the Euclidean distance between the specified point P_l^* of the altered path and the corresponding point P_l' on the reference contour (unaltered path). We say that the two points P_l^*, P_l' are correspondent if the length of the reference path and of the altered path are divided by the two points in exactly the same ratio, respectively. The MAE metric described by (11) is particularly compelling for practical use because it facilitates standardized comparisons of candidate trajectories with the reference path. Its value is higher when the unpredictability of the altered path is higher.

As expected, MAE, computed in L = 1440 points (12 laps, with 120 points per lap), increases linearly with α, as depicted in Figure 10.

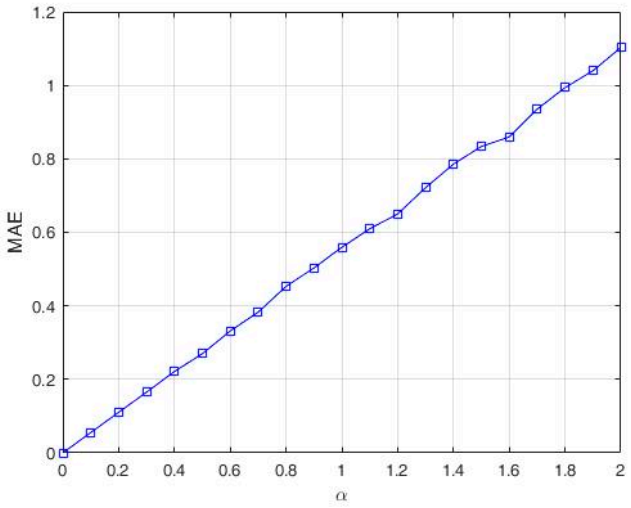

Figure 10. MAE metric for different values of α.

In our case, where $\alpha = 1.7$, the value for the MAE metric obtained through simulations is 0.9352.

While MAE characterizes the long-term unpredictability of the path, the other three metrics that we employed (ANDC, MAA and RA) can provide valuable insights about protean behavior feature of the path. For our method, the average number of direction changes (ANDC) in a given time t is a direct result of selecting the t_{max} and α parameters, while the range of angles (RA) for direction changes derives directly from the design of the method. In the specific conditions taken into account in the first case study, $ANDC$ = 0.4024 direction changes per second and $RA = (-\pi, \pi]$ radians.

Another metric that we used to evaluate the short-term unpredictability (i.e., the main characteristic of protean behavior) is the mean absolute angle (MAA) for direction changes and can be computed using the following equation:

$$MAA = \frac{1}{K-1} \sum_{k=1}^{K-1} |\vartheta_{k+1} - \vartheta_k|, \tag{12}$$

where K is the number of the segments in the altered path, while ϑ_k is the angle between a segment and the x-axis. This metric reflects the uncertainty in predicting the next direction on which the robot will evolve. For the trajectory depicted in Figure 7, we obtained MAA = 0.7815 radians.

5.2. Second Case Study

This case study is aimed to analyze the proposed AWG-based method in the presence of fixed obstacles and also to compare this method with the one based on fictive temporary obstacles (FTOs) generation [35]. For this, we considered a square workspace (12 × 12 square meters), where we marked a reference contour described by four initial waypoints having the following coordinates: $P_1(1, 1)$, $P_2(1, 11)$, $P_3(11, 11)$ and $P_4(11, 1)$. The mobile robot is moving with a constant speed of 1 m/s and uses a bug-like obstacle avoiding mechanism (i.e., TangentBug) based on an obstacle detection sensor having a range of 0.5 m.

In order to study the proposed path-planning method in a more complex environment, we placed a set of fixed obstacles having various shapes and sizes inside the workspace (Figure 11). In this context, as discussed at the end of Section 4, the robot must drop any additional waypoint that falls inside the obstacles. This procedure is very simple if the robot knows the exact position, shape and size of the obstacles. In the case that the obstacles are unknown, we can pursue the following procedure: an additional waypoint is left aside if it is not reached in a given time (e.g., two times higher than the time to reach it in an environment without obstacles).

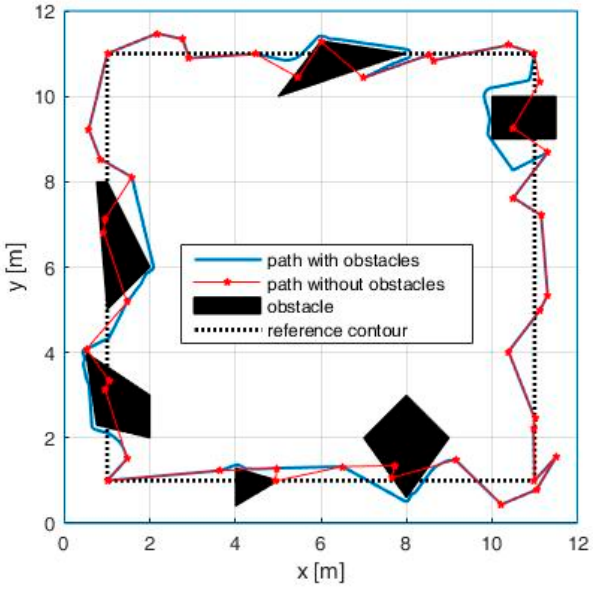

Figure 11. Impact of the obstacles.

From Figure 11, we can see that, in the proximity of small size obstacles, the unpredictability of the trajectory remains high, while, in the case of large obstacles, where more additional waypoints are dropped, the unpredictability gets lower, the robot being forced to take a circumnavigating path around obstacles.

The second objective of this case study is to compare the proposed AWG-based method with the one based on FTOs, which adopts a different strategy to impart protean behavior to the patrol robot. This strategy uses the intrinsic obstacle avoidance mechanism of the robot not only to circumvent real obstacles but also to avoid a random-looking series of fictive and temporary obstacles. In this case study, the two robot's paths are generated as follows: (i) the path generated using FTOs considered the following parameters of the algorithm: a FTO appearance time between 2 and 20 s, a FTO active time between 0.3 and 2.5 s and the initialization values $x_0 = 0.18$ and ; and (ii) the AWG-based robot path uses $\alpha = 1.3$ to assure the same number of direction changes per lap (ANDC = 0.4024).

In the first scenario, we considered an obstacle-free workspace and we represented the two misleading paths generated by both methods (Figure 12). We computed the mentioned metrics for the two paths obtaining the following results: for the AWG-based path, we have MAE = 0.7120, $RA = (-\pi,$

π] and $MAA = 0.7854$, while, for the FTO-based path, we have $MAE = 1.5769$, $RA = [-\pi/2, \pi/2]$ and $MAA = 0.3681$. These results show two metrics favorable for the AWG-based method and one metric (MAE) favorable for the FTO-based method, while the fourth metric is the same ($ANDC = 0.4024$) for both methods. Overall, based on the values previously presented for the four considered metrics and based on the visual analysis of the simulated trajectories, the AWG based method generates a path, which better emulates the zig-zag movements that characterizes the protean behavior due to two reasons: (i) the path has sudden and more abrupt direction changes; and (ii) path segments with go-backward components are also possible.

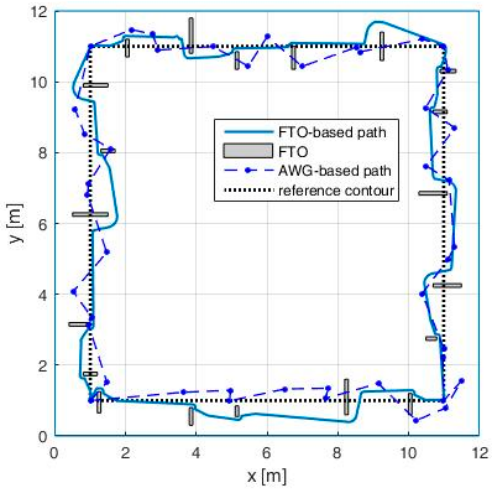

Figure 12. Impact of FTOs above robot's path.

In the second scenario, the two methods are compared in a workspace containing different fixed obstacles (Figure 13). While for small size obstacles the AWG-based method is still the best variant, for large obstacles, the FTO-based method offers greater path unpredictability (the FTO are applied even when the robot circumnavigates an obstacle), so a mix between the two methods can be a feasible solution for these situations.

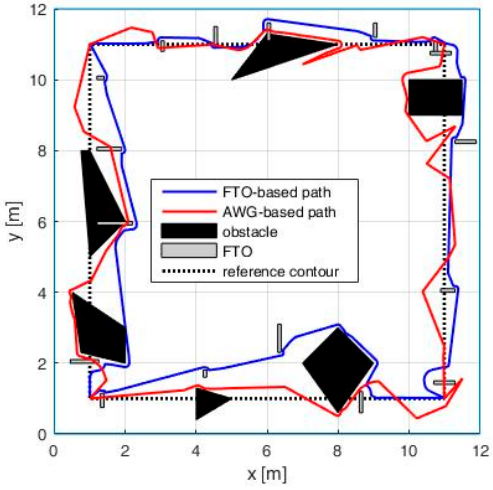

Figure 13. Impact of the obstacles on the path.

6. Conclusions

The paper provides a new bio-inspired method to generate deceptive paths for mobile robots accomplishing patrolling missions under adversarial threat. Using the 2D Arnold cat map as a primary

source of positional entropy, the reference path is altered without compromising the overall goals of the patrolling mission. The obtained misleading trajectory emulates an unpredictable zig-zag motion that is similar to the one developed by individual pray animals when chased by predators. Compared with the method based on fictive-temporary obstacle generation, the present method offers superior results in environments without or with small/medium size obstacles. Furthermore, the paper suggests the idea of mixing the two methods to cope with complex environments where large obstacles can also be encountered.

Author Contributions: All the authors have equally contributed to this work. Conceptualization—D.-I.C., O.B. and C.V.; Methodology—D.-I.C., O.B., C.V. and C.-D.C.; Software—D.-I.C., O.B. and C.-D.C.; Supervision—D.-I.C.; Validation—C.-D.C.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Shannon, C.E. A Mathematical Theory of Communication. *Bell Syst. Tech. J.* **1948**, *27*, 379–423, 623–656. [CrossRef]
2. Cover, T.M.; Thomas, J.A. *Elements of Information Theory*; Wiley: Hoboken, NJ, USA, 2012.
3. Pal, N.R.; Pal, S.K. Entropy: A new definition and its applications. *IEEE Trans. Syst. Man Cybern.* **1991**, *21*, 1260–1270. [CrossRef]
4. Roberts, S.; Guilford, T.; Rezek, I.; Biro, D. Positional entropy during pigeon homing I: Application of Bayesian latent state modelling. *J. Theor. Biol.* **2004**, *227*, 39–50. [CrossRef] [PubMed]
5. Guilford, T.; Roberts, S.; Biro, D.; Rezek, I. Positional entropy during pigeon homing II: Navigational interpretation of Bayesian latent state models. *J. Theor. Biol.* **2004**, *227*, 25–38. [CrossRef] [PubMed]
6. Herbert-Read, J.E.; Ward, A.J.; Sumpter, D.J.; Mann, R.P. Escape path complexity and its context dependency in Pacific blue-eyes (*Pseudomugil signifer*). *J. Exp. Biol.* **2017**, *220*, 2076–2081. [CrossRef] [PubMed]
7. Edmunds, M. *Defence in Animals: A Survey of Anti-Predator Defences*; Longman Publishing Group: Harlow, UK, 1974.
8. Chance, M.R.A.; Russell, W.M.S. Protean displays: A form of all aesthetic behavior. *J. Zool.* **1959**, *132*, 65–70.
9. Humphries, D.A.; Driver, P.M. Protean defense by prey animals. *Oecologia* **1970**, *5*, 285–302. [CrossRef] [PubMed]
10. Shahaf, E.; Eilam, D. Protean behavior under barn-owl attack: Voles alternate between freezing and fleeing and spiny mice flee in alternating patterns. *Behav. Brain Res.* **2004**, *155*, 207–216.
11. Domenici, P.; Booth, D.; Blagburn, J.M.; Bacon, J.P. Cockroaches keep predators guessing by using preferred escape trajectories. *Curr. Biol.* **2008**, *18*, 1792–1796. [CrossRef] [PubMed]
12. Driver, P.M.; Hamphries, D.A. *Protean Behavior: The Biology of Unpredictability*; Clarendon Press: Oxford, UK, 1988.
13. Nahin, P.J. *Chases and Escapes: The Mathematics of Pursuit and Evasion*; Princeton University Press: Princeton, NJ, USA, 2012.
14. Li, W. A dynamics perspective of pursuit-evasion: Capturing and escaping when the pursuer runs faster than the agile evader. *IEEE Trans. Autom. Control* **2017**, *62*, 451–457. [CrossRef]
15. Basilico, N.; Gatti, N.; Amigoni, F. Patrolling security games: Definition and algorithms for solving large instances with single patroller and single intruder. *Artif. Intell.* **2012**, *184*, 78–123. [CrossRef]
16. Hernández, E.; Barrientos, A.; Del Cerro, J. Selective Smooth Fictitious Play: An approach based on game theory for patrolling infrastructures with a multi-robot system. *Expert Syst. Appl.* **2014**, *41*, 2897–2913. [CrossRef]
17. Agmon, N.; Kaminka, G.A.; Kraus, S. Multi-robot adversarial patrolling: Facing a full-knowledge opponent. *J. Artif. Intell. Res.* **2011**, *42*, 887–916.
18. Song, J.; Gupta, S.; Hare, J. Game-theoretic cooperative coverage using autonomous vehicles. In Proceedings of the IEEE Oceans 2014, St. John's, NL, Canada, 14–19 September 2014.
19. Portugal, D.; Rocha, R.P. Multi-robot patrolling algorithms: Examining performance and scalability. *Adv. Robot.* **2013**, *27*, 325–336. [CrossRef]

20. Tews, A.D.; Sukhatme, G.S.; Mataric, M.J. A multi-robot approach to stealthy navigation in the presence of an observer. In Proceedings of the 2004 IEEE International Conference on Robotics and Automation, New Orleans, LA, USA, 26 April–1 May 2004.
21. Araiza-Illan, D.; Dodd, T.J. Bio-inspired autonomous navigation and escape from pursuers with potential functions. In *Advances in Autonomous Robotics, Proceedings of the TAROS 2012, Bristol, UK, 20–25 August 2012*; Herrmann, G., Ed.; Springer: Berlin, Germany, 2012.
22. Chung, T.H.; Hollinger, G.A.; Isler, V. Search and pursuit-evasion in mobile robotics. *Auton. Robot.* **2011**, *31*, 299. [CrossRef]
23. Agmon, N. Robotic strategic behavior in adversarial environments. In Proceedings of the 26th International Joint Conference on Artificial Intelligence, Melbourne, Australia, 19–25 August 2017.
24. Strogatz, S.H. *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering*; Avalon Publishing: New York, NY, USA, 2014.
25. Zang, X.; Iqbal, S.; Zhu, Y.; Liu, X.; Zhao, J. Applications of chaotic dynamics in robotics. *Int. J. Adv. Robot. Syst.* **2016**, *13*, 60. [CrossRef]
26. Martins-Filho, L.S.; Macau, E.E.N. Patrol mobile robots and chaotic trajectories. *Math. Probl. Eng.* **2007**, *2007*. [CrossRef]
27. Curiac, D.I.; Volosencu, C. Chaotic trajectory design for monitoring an arbitrary number of specified locations using points of interest. *Math. Probl. Eng.* **2012**, *2012*. [CrossRef]
28. Curiac, D.I.; Volosencu, C. Developing 2D Chaotic Trajectories for Monitoring an Area with Two Points of Interest. In Proceedings of the 10th WSEAS International Conference on Automation & Information—ICAI 2009, Prague, Czech Republic, 23–25 March 2009.
29. Curiac, D.I.; Volosencu, C. A 2D chaotic path planning for mobile robots accomplishing boundary surveillance missions in adversarial conditions. *Commun. Nonlinear Sci.* **2014**, *19*, 3617–3627. [CrossRef]
30. Volos, C.K.; Kyprianidis, I.M.; Stouboulos, I.N. Experimental investigation on coverage performance of a chaotic autonomous mobile robot. *Robot. Auton. Syst.* **2013**, *61*, 1314–1322. [CrossRef]
31. Volos, C.K.; Kyprianidis, I.M.; Stouboulos, I.N. A chaotic path planning generator for autonomous mobile robots. *Robot. Auton. Syst.* **2012**, *60*, 651–656. [CrossRef]
32. Li, C.; Wang, F.; Zhao, L.; Li, Y.; Song, Y. An improved chaotic motion path planner for autonomous mobile robots based on a logistic map. *Int. J. Adv. Robot. Syst.* **2013**, *10*, 273. [CrossRef]
33. Li, C.; Song, Y.; Wang, F.; Wang, Z.; Li, Y. A bounded strategy of the mobile robot coverage path planning based on Lorenz chaotic system. *Int. J. Adv. Robot. Syst.* **2016**, *13*, 107. [CrossRef]
34. Nakamura, Y.; Sekiguchi, A. The chaotic mobile robot. *IEEE Trans. Robot. Autom.* **2001**, *17*, 898–904. [CrossRef]
35. Curiac, D.I.; Volosencu, C. Imparting protean behavior to mobile robots accomplishing patrolling tasks in the presence of adversaries. *Bioinspir. Biomim.* **2015**, *10*, 056017. [CrossRef] [PubMed]
36. Ng, J.; Bräunl, T. Performance comparison of bug navigation algorithms. *J. Intell. Robot. Syst.* **2007**, *50*, 73–84. [CrossRef]
37. LaValle, S.M. *Planning Algorithms*; Cambridge University Press: Cambridge, UK, 2006.
38. Özkaynak, F. A novel method to improve the performance of chaos based evolutionary algorithms. *Optik* **2015**, *126*, 5434–5438. [CrossRef]
39. Arnold, V.I.; Avez, A. *Ergodic Problems of Classical Mechanics*; Benjamin: New York, NY, USA, 1968.
40. Curiac, D.I.; Volosencu, C. Path planning algorithm based on Arnold cat map for surveillance UAVs. *Def. Sci. J.* **2015**, *65*, 483–488. [CrossRef]
41. Chen, Y.L.; Yau, H.T.; Yang, G.J. A Maximum Entropy-Based Chaotic Time-Variant Fragile Watermarking Scheme for Image Tampering Detection. *Entropy* **2013**, *15*, 3170–3185. [CrossRef]
42. Chen, G.; Mao, Y.; Chui, C.K. A symmetric image encryption scheme based on 3D chaotic cat maps. *Chaos Solitons Fractals* **2004**, *21*, 749–761. [CrossRef]
43. Curiac, D.I.; Volosencu, C. A generic method to construct new customized-shaped chaotic systems using the relative motion concept. *Nonlinear Anal. Model. Control* **2016**, *21*, 413–423.
44. Kamon, I.; Rivlin, E. Sensory-based motion planning with global proofs. *IEEE Trans. Robot. Autom.* **1997**, *13*, 814–822. [CrossRef]
45. Kamon, I.; Rivlin, E.; Rimon, E. Range-sensor based navigation in three dimensions. In Proceedings of the 1999 IEEE International Conference on Robotics & Automation, Detroit, MI, USA, 10–15 May 1999.

46. Rashid, A.T.; Ali, A.A.; Frasca, M.; Fortuna, L. Path planning with obstacle avoidance based on visibility binary tree algorithm. *Robot. Auton. Syst.* **2013**, *61*, 1440–1449. [CrossRef]
47. Arena, P.; De Fiore, S.; Fortuna, L.; Patané, L. Perception-action map learning in controlled multiscroll systems applied to robot navigation. *Chaos* **2008**, *18*, 043119. [CrossRef] [PubMed]
48. Kamon, I.; Rimon, E.; Rivlin, E. Tangentbug: A range-sensor-based navigation algorithm. *Int. J. Robot. Res.* **1998**, *17*, 934–953. [CrossRef]
49. Choset, H.M.; Lynch, K.M.; Hutchinson, S.; Kantor, G.A.; Burgard, W.; Kavraki, L.E.; Thrun, S. *Principles of Robot Motion: Theory, Algorithms, and Implementation*; MIT Press: Cambridge, MA, USA, 2005; ISBN 9780262033275.