Department of Information Science and Technology

A Cooperative Active Perception approach for Swarm Robotics

Pedro Sousa Romano

A Dissertation presented in partial fulfillment of the Requirements for the Degree of

Master in Telecommunications and Computer Engineering

Supervisor

Prof. Dr. Luís Miguel Nunes, Assistant Professor

ISCTE-IUL

Co-Supervisor

Prof. Dr. Sancho Moura Oliveira, Assistant Professor

ISCTE-IUL

January 2018

"To a robot, the world is a sea of ambiguity, in which it will sink or swim depending on the robustness of its perceptual abilities."

Paul Fitzpatrick

Resumo

Mais de um século após a robótica moderna ter surgido, ainda nos deparamos com um cenário onde a maioria do trabalho executado por robôs é pré-determinado, ao invés de autónomo. Uma forte compreensão do ambiente é um dos pontos chave para a autonomia, permitindo aos robôs tomarem decisões corretas baseadas no ambiente que os rodeia.

Abordagens mais clássicas para obter controladores de robótica são baseadas na especificação manual, mas tornam-se menos apropriadas à medida que a complexidade aumenta. Métodos de inteligência artificial como algoritmos evolucionários foram introduzidos para obter controladores de robótica através da otimização de uma rede neuronal artificial para uma função de fitness que mede a aptidão dos robôs para resolver uma determinada tarefa.

Neste trabalho, é apresentada uma nova abordagem para perceção do ambiente por um enxame de robôs, com um modelo de comportamento baseado na identificação cooperativa de objetos que circulam no ambiente, seguida de uma atuação baseada no resultado da identificação. Os controladores são obtidos através de métodos evolucionários. Os resultados apesentam um controlador com uma alta taxa de identificação e de decisão.

Segue-se um estudo sobre o escalonamento da abordagem a múltiplos ambientes. São feitas experiencias num ambiente terrestre, marinho e aéreo, bem como num contexto ideal, ruidoso e híbrido. No contexto híbrido, diferentes samples da evolução ocorrem em diferentes ambientes. Os resultados demonstram a forma como cada controlador se adapta aos restantes ambientes e concluem que a evolução híbrida foi a mais capaz de gerar um controlador robusto e transversal aos diferentes ambientes.

Palavras-chave: Robótica evolucionária, Sistemas multi-robô, Cooperação, Perceção, Identificação de objetos, Inteligência artificial, Aprendizagem automática, Redes neuronais, Múltiplos ambientes.

Abstract

More than half a century after modern robotics first emerged, we still face a landscape in which most of the work done by robots is predetermined, rather than autonomous. A strong understanding of the environment is one of the key factors for autonomy, enabling the robots to make correct decisions based on the environment surrounding them.

Classic methods for obtaining robotic controllers are based on manual specification, but become less trivial as the complexity scales. Artificial intelligence methods like evolutionary algorithms were introduced to synthesize robotic controllers by optimizing an artificial neural network to a given fitness function that measures the robots' performance to solve a predetermined task.

In this work, a novel approach to swarm robotics environment perception is studied, with a behavior model based on the cooperative identification of objects that fly around an environment, followed by an action based on the result of the identification process. Controllers are obtained via evolutionary methods. Results show a controller with a high identification and correct decision rates.

The work is followed by a study on scaling up that approach to multiple environments. Experiments are done on terrain, marine and aerial environments, as well as on ideal, noisy and hybrid scenarios. In the hybrid scenario, different evolution samples are done in different environments. Results show the way these controllers are able to adapt to each scenario and conclude a hybrid evolution is the best fit to generate a more robust and environment independent controller to solve our task.

Keywords: Evolutionary Robotics, Multirobot systems, Cooperation, Perception, Object identification, Artifical intelligence, Machine Learning, Neural networks, Multiple environments.

Acknowledgements

I would like to thank my supervisor Professor Luís Nunes and co-supervisor Professor Sancho Oliveira for their dedication and knowledge, not only throughout this thesis but also with previous investigations and projects along my academic journey.

I would also like to thank all my friends and colleagues for their friendship, motivation and constant support. I will be forever grateful for how much I have grown personally and intellectually alongside them these last few years.

A special thanks also goes to my University, ISCTE-IUL, and all the teachers and services that were always available and supportive.

Finally, I thank my family and specially my parents Filomena and Jorge for their support, love, and for allowing me to have the privilege of completing this higher education degree.

Contents

List of Figures

List of Tables

List of Algorithms

Abbreviations

AI	**A**rtificial **I**ntelligence (see page 2)
ANN	**A**rtificial **N**eural **N**etwork (see page 2)
CTRNN	**C**ontinuous-**T**ime **R**ecurrent **N**eural **N**etwork (see page 9)
EA	**E**volutionary **A**lgorithm (see page 2)
ER	**E**volutionary **R**obotics (see page 2)
UAV	**U**nmanned **A**erial **V**ehicle (see page 35)

Chapter 1

Introduction

Ever since its first steps around the 1940's, modern robotics have evolved and we are now able to produce machines that can replace humans in many tasks. Still, most of the work that robots are now broadly used for is predetermined, like in factories and manufacturing, and do not involve great levels of autonomy, or many degrees of freedom. The penetration of fully autonomous robots in society is still scarce. One of the key factors that make up this challenge is environment perception. In order to behave autonomously, the robot needs to make a wide variety of decisions that have to be supported by a great understanding of the environment surrounding it [1].

In nature, perception is often linked with psychological factors rather than just physiologic [2]. Many times, the reaction to a stimulus depends on the context the animal finds itself in. Also, learning and experience build a more solid understanding of the environment in the long term, changing the reaction to the same stimulus over time. Although many of these approaches are unfeasible in computer models, inspiration in nature while developing robotic approaches often leads to natural and simple solutions.

"Machine Perception" is a term used to describe the capability of a machine to interpret data much like humans use their senses to perceive the world around it. Although humans perceive and interpret the world in ways that far surpass today's technological capabilities, it is a good benchmark to set. A good level of perception will ultimately boost the level of situation awareness, greatly improving the chances of making a good decision.

Classic methods for synthesizing robotic controllers are based on the manual specification of it's behavior. For greater levels of complexity, manually specifying all possible use cases and scenarios a robot may encounter gets specially demanding. This has motivated the application of artificial intelligence (AI) to synthesize robotic controllers. In the 1990's, the first experiments were conducted using evolutionary computation (a subfield of AI and machine learning) to synthesize robotic controllers with evolutionary algorithms (EAs). This approach started having promising results [3, 4] as the evolutionary robotics (ER) field of study started to gain shape. Using this approach, an initial random controller is optimized through several generations. At each generation, a population of candidate solutions is tested and the best performing solutions are mutated and passed on to the next generation. With this method, we get an incrementally better controller at each generation as we let evolution take care of the controller specification, much like Darwin suggests evolution takes place in nature.

A common framework for robotic controllers is an artificial neural network (ANN). This approach is inspired by the way the human brain works, with computer models of axons and neurons. Each neuron usually has an activation function that allows or inhibits the propagation of the information to the other neural units. One of the main advantages of the ANN framework applied to robotic controllers is the resistance to noise [5], introduced for example by the normal imperfections of real-world hardware (sensors). The ANN framework is also a natural fit for robotics, with its layer architecture allowing for a direct mapping of the sensors to the input layer and the actuators to the output layer. Sensor activation in ANN's are usually represented by a value in specific range. For example, when detecting an obstacle, a sensor can feed into the network a value in the range [0,1] where 1 would refer to a very close object and 0 would be provided when no obstacle is in sight.

Environment perception in robotics is a natural evolution driven by the need of making robots ever more autonomous and intelligent. Different approaches on this subject have been studied over the years, based on voice [1], vision [6, 7, 8] and touch [9] to perceive the environment. Furthermore, extracting relevant behavior from a noisy perception is a major challenge that is addressed in studies like [7, 8, 10]. Investigation on this subject although very diverse in the means of perceiving and acting upon the environment, all fall primarily in the same setting: a terrain environment.

With the proliferation of devices like drones and the expansion of robotic applications, it's important to explore different environments and create solutions that can be applied to multiple scenarios. In particular, this work will study terrain, aerial and marine environments and the challenges that arise in both developing cooperative active perception capabilities for swarms that are scalable to multiple environments and the new challenges introduced by each of the environments' singularities.

In the scope of this dissertation, perceiving the environment can be described as the identification of objects, it's features and further classification. Upon the results of that classification, the robot can act on the environment, changing its state. The perception of each robot is shared with the team-members in the field of sight. This aggregates as a cooperative active perception approach to swarm robotics.

We will focus in a task where a swarm of robots navigates through an environment with unidentified objects flying by. These objects carry a set of features, each of which can be observed from a different viewpoint. The robots have three goals: (i) identifying all the features of the objects, (ii) catching the objects that fall in a certain category defined by the presence of a specific set of features and (iii) keeping a formation like distribution on the environment, simulating a patrolling behavior inside the arena. Although collective object identification is not a novel issue, the introduction of marine and aerial singularities and the expectation of creating an environment independent solution has not yet been studied and can have profound applications, from marine surveillance operations to aerial forest fires detection.

1.1 Objectives

The main objective of this dissertation is to present a novel approach to the development and evolution of robotic controllers capable of collectively perceiving and acting upon an environment, in a way that is transversal to many types of environments (i.e. terrain, maritime and aerial).

In summary, the key objectives are:

- Develop a cooperative active perception approach that is scalable to different types of environments and its singularities.
- The demonstration of the approach successfully working on a simulation environment with better chances of real-world transferability.

1.2 Research Challenges and Contribution

To achieve the global goal of this dissertation, multiple challenges must be considered in various areas corresponding to each of the modules of the task we will be focusing on. The module of object identification poses various learning challenges, while the sensing also creates challenges in terms of vision and overcoming noisy perception when confronted with the singularities of various environments that are less linear than terrain environments.

Throughout this work, we will address two main research questions:

- Can EAs generate a cooperative active perception approach to swarm robotics?
- Can we develop a uniform solution that is scalable to different types of environments and its singularities?

1.2.1 Other Scientific Contribution

During our research, we conducted a study related to the robustness of neuroevolved swarm controllers, whose results will bring relevant insights to this study (see Chapter 4.2.3). This contribution has resulted in one conference publication:

- P. Romano, L. Nunes, A. L. Christensen, M. Duarte, S. M. Oliveira, "Genome Variations: Effects on the robustness of neuroevolved swarm controllers", in Proceedings of the Iberian Conference on Robotics (ROBOT), Springer, Berlin, Germany, 2015, pp. 309-319.

1.3 Research Method

The Design Science Research method will be followed to conduct this research, as our focus is in the development and functional performance of the artifact that will be created. We propose to create a method and an instantiation of the artifact. This study will follow a problem centered approach, as the initial research questions were derived from the study and observation of the initial problem.

To conduct the evaluation, simulation methods will be used: the artifact will be executed in a simulation environment with artificial data. The research communication will be carried out thought scientific publications.

1.4 Structure of the Dissertation

In Chapter 2, we start with a brief history on EAs, its application in robotics and the common framework used for structuring the controllers, followed by a review of studies on cooperative active perception approaches for swarm robotics. In Chapter 3, we present our approach for a cooperative active perception control system, detail the development iterations and discuss the controller performance results. Chapter 4 will focus in testing and adapting the controller to multiple environments. In Chapter 5, global conclusions of this thesis are presented.

Chapter 2

State of the Art

From health to space exploration, robots are widely used across several industries. As it's popularity increases and use cases become more diverse, one of the most complex challenges in this discipline today revolves around robot autonomy and intelligent behavior. Sensing the environment is one of the key features to enable a fully autonomous behavior. To successfully develop a controller with these capabilities, several problems need to be considered, in multiple areas: environment perception, object recognition and computer vision.

Furthermore, multirobot systems are becoming common in tasks like autonomous surveillance and monitoring. In these cases, additional challenges (and opportunities) arise in the inter-robot cooperation for environment perception and decision making.

In section 2.1, we start with an overview of ER, the technique that will be used in synthesis of the robotic controllers developed throughout this dissertation. In section 2.2, we review various approaches studied for solving the cooperative active perception challenges in swarm robotics for autonomous robots.

2.1 Evolutionary Robotics

2.1.1 Evolutionary Computation

Evolutionary computation is a sub-field of artificial intelligence in which EAs are used. These algorithms are inspired on biological mechanisms, following the same principles as the natural evolution described by Darwin. By creating populations of potential solutions, evaluating and mutating them, solutions can be refined, making this a global optimization method. The use of these principles for automated problem solving started in the 1950's.

To conduct the evolution, the information for each individual is encoded in a genome, like in biology. Mutation and cross-over is then applied to create the descendants. Then, the evaluation phase takes place. A previously set fitness function measures the quality of the solution. To create the next population, a known approach is to integrate the top performing solutions (highest fitness) in the next generation together with mutations of them. This process cycles and we get back to the mutation and cross-over phase. In the first generation, a common criterion is to generate a random genome. The fitness function plays one of the most important roles in the evolution, defining the balance of the objectives to be reached in order to get the most adequate solution after a couple generations.

2.1.2 Artificial Neural Networks (ANN)

ANNs are the most common framework of ER controllers. This approach is inspired by the way the human brain processes information, like biological neurons, and was first introduced by Warren McCulloch and Walter Pitts in 1943 [11]. They created a computational model for a neural network based on mathematics, stating that the nervous activities, neural events and relations can be described in terms of propositional logic. Further studies started applying the model to the development of robotic solutions [12].

A typical neural network includes five components: (i) the input layer, (ii) the hidden layer, (iii) the output layer, (iv) the weighted connections between each of the previous components and (v) the activation function that converts the input to the output in each of the nodes (neurons). The weighted connections as well as the activation function for the neurons are the main parameters that mold an abstract ANN framework to solve a concrete problem. When EAs are used, these parameters are obtained via the global optimization methods characteristic of this approach. This process replaces the manual specification of the solution and it is the main advantage of using this method.

Early approaches were often based on a specific type of ANN, a discrete time neural network. Continuous-time recurrent neural networks (CTRNN) were later introduced by Joseph Chen in 1998 with appealing results [13], filling the gap of the discrete time neural network's lack of temporal dynamics, like short term memory.

2.1.3 Applications in Robotics

ER comes as a natural concretization of EAs to synthesize robotic controllers. These methodologies started emerging in the 1990's. In 1992, M. Anthony Lewis describes the development of a complex motor pattern generator to control a walking robot in [3], where an ANN is used with weights determined by EAs. In the same year, Dave Cliff presents results that demonstrate the success of using same approach to create an accurate simulation model of a visually guided robot [4]. Even when the fitness function didn't imply the monitoring of visual inputs, the evolution developed those capabilities to solve the task. The author considers the results sufficiently promising of future success in the area.

Although the approach has proven successfully in evolving creative solutions for simple behaviors like foraging, formation, aggregation, etc, one of the biggest challenges in the area is scaling up the approach to more complex tasks, mainly due to the bootstrapping problem, where the goal is so hard/distant that all individuals in the first generation perform equally bad not allowing evolution to start. Transferring the robotic controllers from simulation to real environments (crossing the reality gap) is another big challenge. These difficulties dissipated the attention of this approach from the engineering end of the field.

In 1194, S. Nolfi suggests sampling the real environment through real sensors and actuators on a physical robot to extract a simulation model that is closer to the real environment [14], crossing the reality gap. This issue is also addressed by N. Jakobi. In [15], he stated that it is possible to evolve robotic controllers that behave as good in simulation as they do in real hardware by including appropriate levels of noise and carefully designed simulation environments.

D. Floreano points out research directions for this area in the future [16], with a framework to describe and apply fitness functions and an online continuous adaptive controller. The author shows proof that these techniques allow for a better transfer to real hardware and better scaling to complex solutions.

Another approach is presented by C. Hartland in [17]. This work presents an approach for an anticipation enabled controller, where an error estimation of the real-world compared to the simulation environment is fed to the controller and integrated into the simulation process. The method allowed robots to successfully cross the reality gap.

In 2007, M. Eaton presents one of the first application of EAs to develop complex moving patterns of a humanoid robot [18] and successfully transfers the solution to real hardware.

Miguel Duarte conducted a study [19] that introduced a novel methodology for evolving and transferring controllers for complex tasks and solving the bootstrapping problem, and effectively demonstrates the approach in simulation and real hardware. The author suggests that complex tasks can be recursively split into simpler tasks until these are simple enough to be evolved; controllers to manage the activation of these tasks are also evolved. Then, a tree-like composition of simple tasks and its activation controllers make up the solution for the initial complex task.

More recently, K. Scheper presented a robotic solution with a behavior tree framework optimized by an evolutionary algorithm [20]. The framework is compared to the traditional ANN method. Contrary to the ANN architecture, the controller generated by the presented method is understandable and can be manually adapted to cross the reality gap and fit other purposes without having to run the evolutionary process again. This approach presents an increase in performance compared to current methods.

2.2 Cooperative Active Perception

As referred by Paul Fitzpatrick in [1], it is difficult to achieve robust machine perception, but doing so is the key to intelligent behavior. The author also defends an active perception approach, as figure/ground separation is difficult for computer vision. This study captures young infants' learning abilities and applies it to robotics, giving insights on extending a robot's perceptual abilities using experience to improve performance and interpersonal influences to drive a robot's perceptions by an external entity, like a human "caregiver" giving vocal instructions. This author conducted studies using active vision and active sensing for object segmentation, object recognition and orientation sensitivity.

In 2006, Luís Merino [6] used a cooperative perception system for GPS-equipped UAV's to detect forest fires. Here, active vision plays the most important role. A statistical framework is used to reduce the uncertainty of the global objective (the fire position) taking into account each team-member sensor readings and their uncertainty. Besides just cooperative perception, this study extends the teamwork possibilities with heterogeneous teams. Here, teams are heterogeneous in their sensing and processing capabilities. In a real-world scenario, many applications require several sensors that cannot be carried by a single robot, so this approach provides a way to exploit complementarities of different UAV with different attributes and sensors. In this study, some robots in the team have high detection capabilities but also a high false alarm ration, so other robots in the heterogeneous team are used to confirm or discard the information.

Still, the foundation of all this process is profoundly linked to a robust perception, as such, correctly identifying objects. As stated by Q. V. Le in [9], angles in which objects can be viewed are the main variable to increase likeliness of identification. This study produces great results in object identification as the robot is capable of observing the object in many angles until certainty is reached, and was proven to be better than passive observation and random manipulation. The author presents a novel approach to minimize uncertainty of the object, by maximizing mutual information in each step. Given a specific viewpoint, the robot tries to choose the sequence of actions that will give him the most information about the object, with less cost.

To drive the robot's decision making based on an incomplete and noisy perception is another challenge described in 2010 by Matthijs T.J. Spaan in [7] and [8]. The authors propose a Partially Observable Markov Decision Process (POMDP) to develop an integrated decision-theoretic approach of cooperative active perception, as POMDPs *"offer a strong mathematical framework for sequential decision making under uncertainty, explicitly modeling the imperfect sensing and actuation capabilities of the overall system."*. In [7], the authors explore a scenario of cooperation between fixed surveillance cameras and mobile robots where the main task is to reduce uncertainty in it's view of the environment (a global picture for monitoring the system). The authors conclude the POMDP approach provide a good framework for modeling an agent's decision making when considering noisy estimates. In [8], the authors apply the same approach to a Network Robot System (NRS) where a sequential decision making process is used. In this article, a scenario is set up with both fixed and mobile sensors. Mobile sensors give precise observation, but with a cost (moving there and scarce resources). Therefore, decisions on whether it's worth to take robot A or B to a certain place have to be considered. The management of these local decisions until the global objective is reached is done sequentially. Later in 2014, the authors wanted to expand the approach to include information gain and introduced a new type of POMDP, POMDP-IR (Information Reward) in [10]. This new approach extends the action space with actions that return information rewards. This way the approach stays inside the POMDP framework with it's advantages, while able to model information-gain tasks.

Another robot control approach for a perception-driven swarm is presented by Aamir Ahmad in 2013 [21]. Here, the author proposed and implemented a method for a perception-driven multirobot formation control, with a weighted summed term cost function to control multiple objectives. This study was successful in demonstrating that the authors approach enables a team of homogeneous robots to minimize the uncertainty of a tracked object while satisfying other criteria such as keeping a formation. The approach consists of two main modules, a controller and an estimator. The controller is a distributed non-linear model predictive controller (DNMPC) with the control objective of keeping the formation while minimizing uncertainty of the tracked object. The estimator is based on the particle filter localization method that estimates the target position and velocity to enable cooperative target tracking. The novelty of this study is to integrate the controller and estimator modules in the formation control loop. This is done by including both the cooperative target estimate and the formation criteria in the controller (DNMPC) cost function. This allows for the controller module on each robot to perform the optimization after the estimator object position fusion took place, making the optimization complexity constant (not dependent of the number of mobile sensors in the team). Also, the decoupling of the optimization from the estimator makes the approach more reliable in case of sensor or communication failure.

When GPS or other reliable location features are unavailable, inaccurate self-localization can have a strong negative impact in the global performance, as the relative position of objects transposed to the host's frame becomes inaccurate [22]. Andreas Rauch presents an approach in 2013 to reduce the negative impact of inaccurate self-localization, applied to road vehicles. The author considers that the *"association and fusion of data from distinct sources are major challenges in cooperative perception systems"* and that temporal and spatial alignment is crucial for combining the perception of multiple team-members. Deterministic, probabilistic and numerical approaches were compared and the author concludes that the proposed iterative closest point algorithm is more capable of reducing the average error between the objects and the local perceived objects.

Seong-Woo Kim states that fusing data from remote sensors has various challenges [23]. The author focuses on the map merging problem and sensor multimodality between swarm members to successfully extend perception range beyond that of each member's sensors. Compared with cooperative driving without perception sharing, this approach was proven better at assisting driving decisions in complex traffic situations. The author proposes triangulation and map reckoning to get the relative pose of the nodes allowing the information to be properly fused. The approach assumes no common coordinate system making it more robust.

In 2015, Tiago Rodrigues addressed the sensor sharing challenges as well. In [24] the author proposes local communication to share information sensor between neighbors to overcome constraints of each member's local sensors. Triangulation is used to georeference the tracked object. The proposed approach is transparent to the controller, working as a collective sensor. A variant in which the sensors have a memory is also experimented. Both scenarios were able to achieve a much better performance than classic local sensors.

André Dias conducted a study focused on the tracking of 3D objects in a multirobot perspective [25]. The main problems arise as the target dynamically moves in complex environments. Limited sensor field of view and partial observability as well as object occlusion may limit the system performance. The author considers that to achieve better results, different sources of uncertain information need to be treated accordingly and proposes a multi-robot triangulation method as a novel sensor combined with a decentralized stochastic filter, and achieved better results than with the classical probabilistic filters used used to estimate the target position with different sources of uncertain information. The purpose of the study is to solve initialization and data association problems.

Benjamin Burchfiel presents a novel approach to represent 3D objects for robotic interaction [26] that present several advantages to current models. Classic methods don't allow knowledge transfer from previously seen objects, create large databases of information or require manual training with object models (not very scalable to real-world applications). The author develops a Bayesian Eigenobjects representation that naturally facilitates object detection, pose estimation and classification. Once a model is learned, it is not necessary to retain the original object, just a few parameters, creating a small database. Partial object completion is also possible, very useful in real-world scenarios to estimate inaccessible viewpoints.

These techniques present a diverse contribution in terms of the robotic controllers used, and the sensing and actuating capabilities. In most of the cited work, active vision played the central role of the approach [1, 6, 9]. In [6], a statistical framework is used in the controllers to estimate the target position, and perception with heterogeneous teams is tested. In [8, 7, 10], POMDP's were used to model decision making under uncertainty (good for noisy perceptions). The control of multiple objectives in a robotic solution is presented in [21]. Fusing data sensed between multiple nodes also poses challenges studied in [23], and [24] presents a shared sensor solution to the same problem. 3D object representation in robotics is addressed in [26].

The studies presented above develop and test perception solutions centered in the linear terrain environment, with few to no attention given to the development of cooperative active perception systems using ER. The work presented in this thesis differs in proposing a generic solution scalable to multiple types of environments and overcoming the challenges of the environments' singularities, using ER techniques.

Chapter 3

Cooperative Active Perception Control System

In this chapter, we describe an approach for a swarm robotics control system capable of collectively identifying objects and making decisions based on the identification. It's a common approach in robotic perception to unfold the identification of objects as the identification of specific features that build to a known object or class of objects [1, 9]. Our approach follows that direction: the identification of an object is completed when all its key features are seen by at least one of the robots in the team. Those features can be sensed: (i) directly by each team member using it's local sensor and (ii) indirectly through the shared sensor that allows each robot to sense object features being seen by the teammates in sight. From the controller's point of view, there is no distinction between the local and the shared sensing of a feature.

We'll use a task in which a team of robots must collectively identify a set of objects that fly by, and catch the ones that fall into a certain category (have a specific set of features). The performance of the solution will be measured as the percentage of objects identified and correctly caught by the swarm. Using this criteria, a final evaluation will be done to the best controller resulting from the evolutionary process. Results are presented in section 3.4.2. In Chapter 4, this solution will be transfered and adtapted to multiple environments (terrain, marine and aerial), with various techniques that are studied and compared.

3.1 Methodology

The robotic controller will be obtained using the AI methods introduced in section 2.1 and is driven by a CTRNN. The network will be optimized throughout several generations to maximize a fitness function that measures the solution performance.

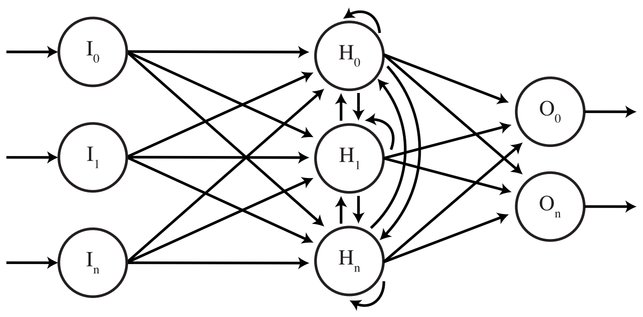

Figure 3.1: Continuous Time Recurrent Neural Network (CTRNN) representation. *I* represents the input layer, *H* the hidden nodes layer and *O* the output layer.

The information from the environment perceived by the robot through its sensors (i.e. distance to objects, local and shared object features, distance to teammates, etc) is mapped to the neural network inputs. A hidden neuron layer is also used, with 5 hidden neurons. The neurons in this layer are connected to each other and to them-selfs, maintaining a state (this allows for short term memory). The output layer of the ANN is connected to the robots actuators, in this case the wheels and the object catch decision (details on the controller architecture will be presented in section 3.4.1). Equation 3.1 describes the network behavior:

$$\tau_i \frac{dH_i}{dt} = -H_i + \sum_{j=1}^{in} \omega_{ji} I_i + \sum_{k=1}^{hidden} \omega_{ki} Z(H_k + \beta_k) \tag{3.1}$$

$$Z(x) = (1 + e^{-x})^{-1} \tag{3.2}$$

where τ_i represents the decay constant, H_i the neuron state and ω_{ji} the strength of the synaptic connection between neurons j and i (the weighted connections represented by the arrows in Fig. 3.1). β represents the bias and $Z(x)$ is the sigmoid function (equation 3.2). *in* represents the total number of inputs and *hidden* the total number of hidden nodes (5 were used). β, τ and ω_{ji} compose the genome that encodes the controller behavior, and are the parameters randomly initialized at the first generation and optimized throughout the evolutionary process, where $\beta \in [-10, 10]$, $\tau \in [0.1, 32]$ and $w_{ji} \in [-10, 10]$. Integrations follow the forward Euler method with an integration step size of 0.2 and cell potentials set to 0 at network initialization.

In our task, robots must cooperate between them to identify objects that appear on the environment. To model restrictions and complexity associated with large objects identification (objects bigger than robots), each robot can only see one feature at a time. This allows to increase the approach scalability to multiple object sizes. With this limitation, cooperation is needed to sense all the features and proceed with the identification. The key is for each individual to balance the local view of one feature with the received shared perception, in order to sense all features of a specific object. The robots should position themselves around an object so that each one is situated in a vantage point that enables it to see one feature directly through its local sensor and all the others indirectly, through the shared sensor that receives the perceptions from nearby teammates. This setup is further explained in section 3.2 and Fig. 3.2. Features don't include information about the object they belong to, so the controller must be able to link them to the correct object, for a successful identification.

When the identification is complete, one of the robots should decide whether or not to catch the object. The objects are divided in two categories: friends and enemies. In the selection phase of the evolutionary process, robots are rewarded for catching enemies and receive a penalty for catching friends. Robots are not given the category of the objects nor the features.

In summary, the key points of the controller behavior are:

- Balancing the local view of one feature with the shared perception received from the nearby teammates to sense all the features and identify the object.
- Extract patterns from the features in an object to deduce its category (friend or enemy) and act based on the conclusion (catch or ignore).

For our experiments, we will use JBotEvolver [27] (`https://github.com/BioMachinesLab/jbotevolver`), a Java-based open-source neuroevolution framework and versatile simulation platform for education and research-driven experiments in ER.

3.2 Experimental Setup

To conduct our experiments, we will use a task in which 8 circular robots with a radius of 5 cm are placed in a 4x4 m bounded environment. The initial position of each robot inside the arena is random, drawn from a uniform distribution. The unidentified objects have a 10 cm radius (twice the size of the robots) and are generated in intervals of 1000 time steps (100 seconds).

Each object carries 4 features distributed around the 4 quadrants of the object's circular perimeter. In the scope of this study, object features are represented by colors, each of which can be observed from a specific viewpoint. The object features are contained in a predefined set of 8 features (4 enemy features and 4 friend features), unknown by the robots. While enemy objects always have the 4 enemy features, friend objects can have a mix of friend and enemy features (up to a max of 2 enemy features). This ambiguity serves a more realistic model and forces robots to evolve a more precise identification process. The order, mix and choice of the features are all uniformly distributed random processes that take place at the generation of each object.

An object is considered identified if all the features are observed for 10 consecutive time steps. Each robot's local features sensor can only view one feature at a time with 2 eyes-like sensor placed at front of it's body. The local perception resulting from this sensor is shared with all the robots inside the teammates sensor radius.

An example of the object identification scenario is depicted on Fig. 3.2. Here, each robot is sensing a different feature of the object with it's front facing local sensor. All robots are inside of each other's range of communication, thus being able to share the local perception. As a result, the 4 robots are able to identify the object, as each of them know all the features.

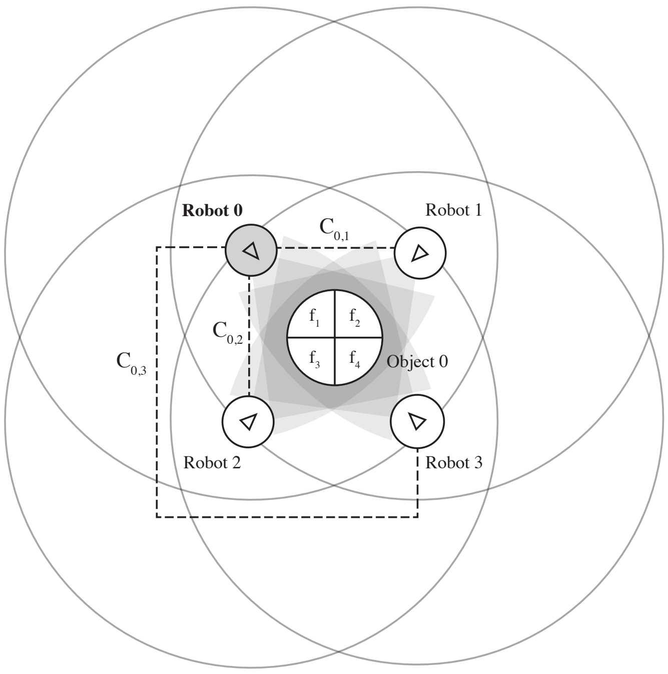

FIGURE 3.2: Schematics of the simulation environment when identifying an object. Object 0 represents the unidentified object, with f1 to f4 representing it's features; robot 0 to robot 3 represent the swarm; grey filled sensors represent the local features sensor of each robot; C represents the communication between each team-member (shared features sensor) and the circular lines represent the field of communication of each robot and it's teammates (radius of robot sensor)

Let's turn our attention to robot 0. With it's front facing local object features sensors, robot 0 knows the object in front of him has the feature f1. Robot 1, 2 and 3 are all in the range of communication of robot 0 and can sense respectively, features f2, f3 and f4 with their own local object features sensor. As such, robot 0 will sense feature f1 thought it's local sensor, but also features f2, f3 and f4 through the shared features sensor. Therefore, robot 0 will be able to sense all the features of the object, identifying it. The same applies to robot 1, 2 and 3. If all the robots know all the features of the object, they should now be able to deduce it's category and decide whether they should catch the object.

3.2.1 Evolutionary Process

To obtain the controller, the evolutionary process was conducted 10 times (evolutionary runs) during 2000 generations. Each generation is composed of 100 individuals, each corresponding to a genome that encodes an ANN. To select the best individuals in a generation, the considered fitness is the average of 30 samples. Each sample is tested during 5000 time steps (500 seconds). For the test, every robot in the swarm has the same genome. After each individual is evaluated, the top 5 are included in the next generation and used to create the remanding 95 individuals of the population: each one of the top individuals generates 19 new individuals by applying gaussian noise to each genome with a probability of 10%.

After the evolutionary process, a post evaluation test was conducted to assert the fitness of the best performing controller (individual) that resulted from the evolution. The fitness resulting from the test is the average of 100 samples with different random seeds. The tests were held during 10000 time steps (1000 seconds), double the time used for the training. This was done to reinforce the statistics.

3.3 Experiments and Results

Multiple iterations of the development were carried out until a solid behavior to solve the initial problem was achieved. Different scenarios, configurations and variables were tested until we reach the final configuration described in section 3.4 and results presented and discussed in section 3.4.2. In this section, the development progress will be described and options taken throughout the process discussed.

3.3.1 Object Identification

In the initial phase of the development, only the object identification behavior was evolved. Robots were awarded for identifying both friends and enemies. At this stage, the team was not required to make any decisions or take any actions based on the identification. The evolution was set to optimize the fitness function described on equation 3.3:

$$F_i = (Objects_{Identified})_i \tag{3.3}$$

where $Objects_{Identified}$ is the number of objects identified. It is important to note how a very simple fitness function was able to evolve such a complex behavior, one of the advantages of using EAs to synthesize robotic controllers.

In the initial phase of the solution development, an architecture was tested in which the object features sensor consisted in 3 distinct readings that, in this scenario, were linked to the RGB values of the color the feature was represented by. This method was discarded as our initial experiments revealed that the evolutionary process was finding unexisting patterns between RGB values, while similar colors still corresponded to completely different features. For this reason and also to further abstract the features implementation, a binary input that indicates whether a specific feature is being seen or not was implemented.

The density of objects on the environment was the main variable to influence the solution performance. A value of 500 time steps between the appearance of a new object was initially used. The ambiguity of having multiple objects on the environment and the possibility of them being too close to each other stopped the robots from understanding the features belonging to each (since no distinction is made) and gave very poor values for object identification ($< 20\%$). A value of 1000 time steps between the generation of each object was able to reduce the ambiguity and reach a very high object identification percentage (95,7%). Nonetheless, scaling the approach with multiple objects to identify at the same time is necessary for a realistic real word solution and will be done in the next iterations of the solution development.

Object speed on the environment also influenced results as the robots needed time to find and position themselves around the objects, specially in our bounded experiment. A value of 0.15 cm/s was found to give good results, allowing each object to stay on the environment long enough to be recognized.

3.3.2 Object Identification and Catching

After a solid object identification controller, the catching actuator was added to the equation. At this point, the evolutionary process is set to evolve a two step controller: (i) object identification and (ii) object class inference based on feature patterns. The catching decision is a direct result from step ii. Solution performance is now measured in terms of the number of enemies caught. Number of enemies identified serves as an auxiliary metric; no attention is now given to the identification of friends.

A formation component was added to the fitness function, to stimulate the robots to evolve a patrolling behavior and spread out inside the arena, maintaining a known distance to each other. The evolution is set to optimize the fitness function described in equation 3.4:

$$F_i = \alpha_i + \beta_i \tag{3.4}$$

$$\alpha_i = \sum_{n=0}^{timesteps} \begin{cases} 10^{-2}, & \text{if ADN} \in [S_r - \frac{S_r}{10}, S_r + \frac{S_r}{10}] \\ -|ADN - S_r| \times 10^{-2}, & \text{else} \end{cases} \tag{3.5}$$

$$\beta_i = \frac{Enemy_{identified}}{5 \times 10^{-3}} + \frac{Enemy_{caught}}{10^{-3}} - \frac{Friends_{caught}}{2 \times 10^{-3}} - \frac{Unidentified_{caught}}{10^{-3}} \tag{3.6}$$

where α_i corresponds to the formation component of the fitness function and β_i corresponds to the object identification component. ADN is the average distance of the robots to it's closest team-mate, S_r is the robot teammates sensor radius. $Enemy_{identified}$ is the total number of enemy objects that were identified during the test, $Enemy_{caught}$ corresponds to the total number of enemy objects caught. $Friends_{caught}$ and $Unidentified_{caught}$ corresponds to the total number of friends and inoffensive objects caught, respectively. The formation component awards the robots for keeping a distance between each other that corresponds to the radius of their teammates sensor (S_r) with an error margin of $\frac{S_r}{10}$. This awards them for dispersing around the environment in search for objects while keeping a known distance to their teammates.

This controller scored: (i) an "enemy identification rate" of 77%, (ii) an "enemy catch rate" of 73% and (iii) a "friends and unidentified objects catch rate" of 2% and 8% respectively. This evaluation is an average of 100 samples during 1000 time steps and was done to the best controller resulting from the evolution.

3.4 Final Controller

To finalize the controller, our attention turned to pushing the boundaries and increasingly raise the complexity of the problem. To make the approach more realistic, the unidentified objects now have more degrees of freedom and can appear from any side of the arena, moving to the opposite side. In 30% of cases, two objects will be on the arena at the same time, also increasing the identification complexity; in the remainder 70% of cases only one object is inside the arena at the same time. The position of the object is randomly assigned when only one object is on the arena at a time and fixed on the bottom and top or left and right portions of the arena when two objects are on the arena at the same time. The possibility of having two objects inside the arena at the same time should force the robots to separate in groups to proceed with the identification. Object speed is now variable, assigned to each object at the moment of creation and corresponding to a random speed drawn from a uniform distribution, between 0.15 and 0.35 cm/s.

This final experiment consolidates the solution and adds enough complexity to state that the approach is realistic and solid, mainly due to the capability of identifying multiple objects on the arena at the same time at variable speeds and from different directions. In the next section, a detailed description of the final controller architecture is shown. Results are presented in section 3.4.2.

3.4.1 Controller Architecture

The controller architecture is composed of 2 actuators and 5 sensors. The actuators consist of (i) two wheels that allow the robot to move inside the environment at a maximum speed of 10 cm/s and (ii) a binary output that represents the categorization decision and allows the robot to catch the objects. The sensors include: (i) a wall sensor for the robot to sense the distance to the boundaries of the arena, (ii) a robot sensor so that each robot senses the distance to the nearest team mate, (iii) an object distance sensor for each robot to sense the distance to the closest object; to identify the objects, an object features sensor (iv) is used, enclosing both the local and the shared perception, allowing the robot to sense the feature in sight, as well as the shared features sensed from the teammates in the field of communication (determined by sensor ii). To allow the robots to disperse in groups when multiple objects are on screen, a robot density sensor (v) is used so each robot knows the percentage of robots that are near, according to the total number of robots in the swarm.

The sensors follow the configuration depicted on Fig. 3.3. Sensors i), ii) and iii) are placed all around the perimeter of the robot and sensor iv) consists in 2 front facing sensors with an eye-like distribution, for a more realistic approach since the perception is based on vision. This also allows the robot to sense the path to reach the object (due to the sensors overlapping at the center).

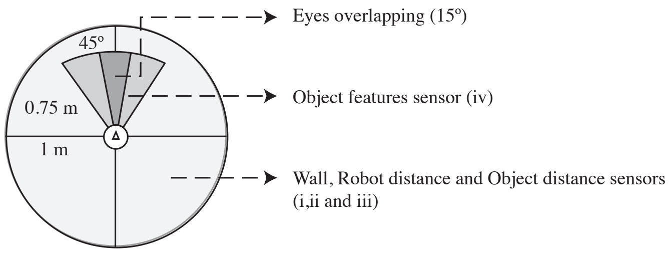
Eyes overlapping (15º)
45º
Object features sensor (iv)
0.75 m
1 m
Wall, Robot distance and Object distance sensors (i,ii and iii)

FIGURE 3.3: Robot sensors representation. 4 sensors with 90° opening angle for sensors i), ii) and iii) and 2 eyes-like sensor with 45° opening angle and 15° of overlapping for sensor iv)

To catch the objects, robots have a binary actuator. When active, the closest object is caught by the robot if situated at a maximum distance of 0.1 m.

All the sensors, actuators and corresponding ANN inputs and outputs are described in Table 3.1.

The reading of each sensor is mapped to the respective neural network input. For distance based sensors (i, ii and iii), the input follows Eq. 3.7

$$i = \frac{range - distance}{range} \tag{3.7}$$

where *range* is the maximum range of the sensor and *distance* is the distance from the robot to the target.

Table 3.1: Controller Architecture: Robot Sensors and Actuators and corresponding ANN Inputs and Ouputs

Sensor	ANN Inputs
i) Wall Sensor Reading in range [0,1] depending on distance to closest wall	4 (total of 4 sensors around the robot each with 90° aperture)
ii) Robot Sensor Reading in range [0,1] depending on distance to closest robot	4 (total of 4 sensors around the robot each with 90° aperture)
iii) Object Distance Sensor Reading in range [0,1] depending on distance to closest robot	4 (total of 4 sensors around the robot each with 90° aperture)
iv) Object Features Shared Sensor Binary readings corresponding to the feature in sight for the closest object	8 (local) + 8 x $\text{N}_{\text{close robots}}$ (shared) (2 local sensors arranged like eyes, with 35° aperture and 10° between the eyes)
v) Robot Density Sensor Reading corresponding to the percentage of robots in sight according to total	1 (1 sensor)
Actuator	ANN Output
i) Two-Wheel Actuator Output in range [0,1] depending on wheel speed	2 (left and right wheel)
ii) Object Catch Actuator Binary output to catch an object	1 (catches closest object at max distance of 0.1 m)

For sensor iv), each preprogrammed feature corresponds to a specific ANN input that can be either 1 or 0 whether that feature is being seen or not. This applies to both the local and the shared inputs, as there is no distinction between the two from the controller standpoint. Sensor v) consists in a single value: the percentage of robots in sight, against the total number of members in the swarm.

3.4.2 Results and Discussion

The evolutionary process was carried out to optimize the fitness function described in equation 3.4. The best controller resulting from the evolution scored an average fitness of 4239 ± 2573. This corresponds to: (i) an "enemy identification rate" of

71%, (ii) an "enemies caught ratio" of 64%, a "friends and unidentified objects caught ratio" of 2% and 7% respectively (seen as false positives of the identification process).

In 80% of the evolutionary runs, the obtained controller successfully solved the task. The robots evolved a behavior in which the team performs a dispersed search around the arena. When one of the teammates senses an enemy object nearby, the robot and it's near teammates get closer and surround it, circumnavigating the object while front-facing it until the identification is complete. When the enemy is identified, one of the teammates decides to catch it, after which they disperse and go back to the search. If the robots sense a friend object, the level of attention given is lower. Less than 4 robots concentrate on the friend objects and they quickly disperse. These are not identified. In 20% of the evolutionary runs, a solution was not found, with the robots evolving a backwards motion behavior that rendered them incapable of visualizing the objects. On average, a good behavior was found around generation 700.

The behavior is also efficient, as only part of the team concentrates on the object that is being identified and no more than 3 robots concentrate on a friend object. The remainder of the team keeps searching for other objects that may appear and could get lost if an unnecessary number of robots concentrated on the identification of only one object.

3.5 Generalizing the Approach

One of the main advantages of the presented controller is it's genericity. ER provides the flexibility of developing solutions based on simple criteria that can be evolved and adapted to multiple setups. The simplicity of the fitness function (number of objects identified and formation component) and the abstraction of the features sensor (binary reading that indicates whether a specific feature is being seen or not) makes this solution easily adaptable to multiple environments, scenarios and types of objects and features.

Object features are a fixed set of possibilities linked to a specific input on the neural network. In our experiments, features are represented by colors, but can be adapted to any other set of features (i.e. object characteristics or sizes), but not limited to. For example, features can also represent the degree of certainty wheels were identified in a moving object, or a certain shape that characterizes a known class of objects. Feature distribution is also scalable. Features can be distributed on multiple objects (identified as a group).

The object catch decision output of the neural network corresponds to the robot categorizing an object as an enemy. In the scope of this project, we considered catching the object as the representation of this decision but, once again, this can be scaled to other approaches (i.e. notifying a central unit, registering the identification).

In Chapter 4, the presented solution will be adapted to a set of environments (terrain marine and aerial), modeled in simulation. Techniques for obtaining a global controller capable of solving the proposed task in all these enviroments will be studied.

Chapter 4

Applying and Adapting the Controller to Multiple Environments

In the previous chapter, we developed a cooperative active perception control system that successfully solved our task. We then scaled the approach introducing random variations of parameters like object speed and possible directions. Although these factors allow the evolution of a more solid, robust and realistic solution, one class of real-world complexity was purposely ignored: external factors.

In the real-world, external factors heavily influence the swarm performance. Furthermore, these factors are mostly singular to each type of environment and of random nature. In our work, up until this point, both the evolutionary process and the evaluation of the controller was conducted under ideal environment conditions. As stated in Chapter 2, the global contribution of this work is not only to present a novel cooperative active perception solution using EAs, but also to fill a gap in the current state of the art: generic solutions, adaptable to multiple environments and it's singularities.

In this chapter, we will model different environments, mainly governed by external conditions that influence the swarm performance. These conditions are models of the main singularities found on each environment. We will test our solution on the different environments with different types of evolution and discuss ways to evolve a global solution that adapts to all of them. Classic methods of obtaining a robust controller capable of crossing the reality gap (i.e. noisy evolution [15]) will also be considered.

Each one of the environments studied presents several relevant practical applications. The terrain environment can simulate obstacles present on complex terrain scenarios; the marine environment can help develop swarms capable of running patrolling and exploration marine tasks; and aerial applications range from aerial drone obstacle avoidance to object detection.

4.1 Modeling the Environments

Three main classes of environments will be modeled in this section: (i) terrain, (ii) maritime and (iii) aerial. These environments represent the majority of conditions the swarm might encounter in a multi-environment real-world scenario. Models of the environments will globally focus on external factors like obstacles, maritime currents or wind. All the agent's solution are built upon the solution presented in Chapter 3.

4.1.1 Terrain Environment

To model the terrain environment, we focused our attention on two main characteristics: (i) accessibility and (ii) irregularities. While terrain irregularities can be handled by the robotic driver and thus don't need to be handled by the controller, accessibility issues like obstacles or object occlusion will benefit from an optimized behavior to solve the task in these conditions. In our model, we included a set of rectangular obstacles distributed around the environment. This creates accessibility difficulties as robots need to deviate from these obstacles to see the objects and proceed with the identification. Robots cannot transpose the obstacles nor see through them.

Inside the same 4x4 m bounded environment with 8 robots and the unidentified objects, a maximum number of 7 obstacles are added, with a minimum value of 25 cm and a maximum value of 65 cm for both width and height. The number of obstacles on the environment, it's positions, width and height are all random processes drawn from a uniform distribution. A sketch of the modeled terrain environment is depicted in Fig. 4.1:

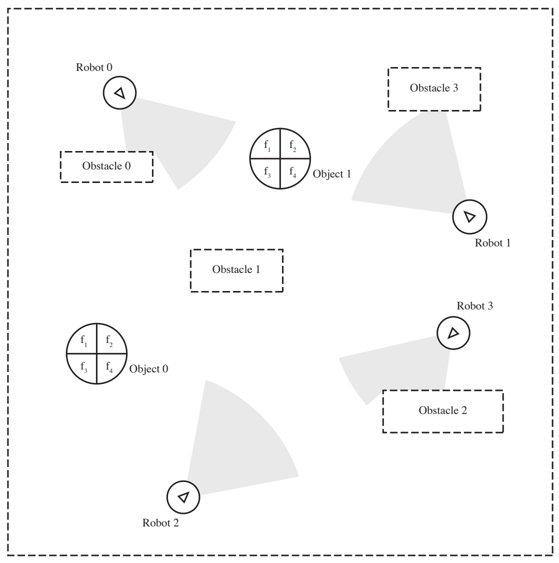

FIGURE 4.1: Schematics of the modeled terrain environment when searching for objects. Object 0 and 1 represents the unidentified objects, with f1 to f4 representing it's features; robot 0 and robot 3 represent the swarm; grey filled sensors represent the local features sensor of each robot; Obstacles 0 to 3 represent the randomly placed objects on the environment with random width and height between [25,65] cm

where only 4 robots and 2 unidentified objects are placed inside the environment (for simplification). With this scenario, we simulate a terrain environment by modeling its increased difficulty in finding and identifying the objects.

4.1.2 Marine Environment

The marine environment model is based on previous studies that successfully obtained swarm robotics controllers capable of crossing the reality gap in a marine environment. Miguel Duarte [28] presents a swarm controller obtained via evolutionary robotics in simulation, capable of working in a real-world marine environment. In this study, the marine environment model used to train the controller in simulation was based on real measurements of the robots motion taken in the water, but without physics simulation and fluid dynamics, as these would make the evolution process too expensive to carry out with reasonable resources. Our model of the marine environment follows this approach and is centered around two main characteristics: (i) a constant dragging current and (ii) robots movement inertia. A sketch of the marine environment is depicted in Fig. 4.2.

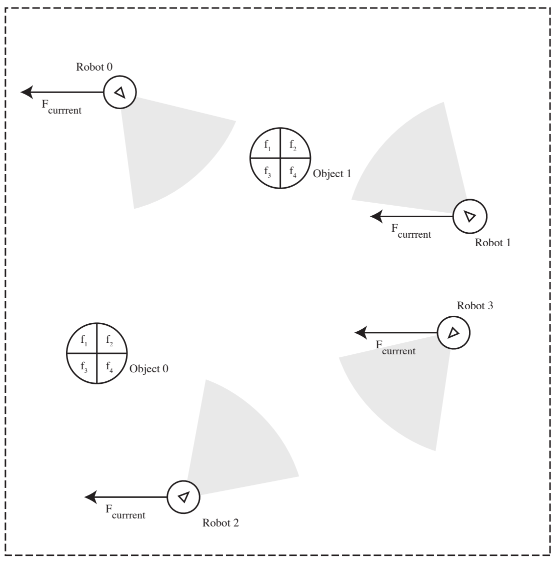

FIGURE 4.2: Schematics of the modeled marine environment when searching for objects. Object 0 and 1 represent the unidentified objects, with f1 to f4 representing it's features; robot 0 to robot 3 represent the swarm; grey filled sensors represent the local features sensor of each robot; F_{current} represents the random constant dragging current applied to the robots' movement with magnitude [-0.1,0.1] cm/s

The constant dragging current is characterized by a vertical and an horizontal magnitude between [-0.1,0.1] cm/s for each axis. The final current is the 2D sum of both axis and can thus drag the robot in any direction. The value of the magnitudes is randomly drawn from a uniform distribution and is fixed throughout each sample (constant current). Inertia in the robots movement is the main characteristic of this environment, with wheel speed for each robot following Algorithm 1:

Algorithm 1 Marine environment robot inertia algorithm

```
if $targetWheelSpeed > previousWheelSpeed + maxIncrementUp$ then
  $wheelSpeed \leftarrow previousWheelSpeed + maxIncrementUp$
else
  if $targetWheelSpeed < previousWheelSpeed - maxIncrementDown$ then
    $wheelSpeed \leftarrow previousWheelSpeed - maxIncrementDown$
  else
    $wheelSpeed \leftarrow targetWheelSpeed$
  end if
end if
```

where $targetWheelSpeed$ represents the speed determined by the controller, $previousWheelSpeed$ represents the previous actual speed of the robot, $wheelSpeed$ represents the actual current wheel speed and $maxIncrementUp$ and $maxIncrementDown$ represents the maximum increment or decrement that the actual speed can suffer from one iteration to the next. $maxIncrementUp$ and $maxIncrementDown$ assume a value of 0.1 m/s. This algorithm is applied to each of the robot's wheels independently.

4.1.3 Aerial Environment

Controlling an Unmanned Aerial Vehicle (UAV) encloses several problems, mostly related to wing gusts and other aerodynamic efforts. Studies like [29, 30, 31] address some of these challenges. In [29], a backstepping approach is used, together with an estimation of the unknown aerodynamic forces to stabilize the position of the vehicle in the presence of aerodynamic forces. In [30], the authors study the influence of wind gusts on the system concluding that it is a crucial problem for real-world outdoor applications, especially on an urban environment. F. Leonard [31] refers that agility, maneuverability and the capability of operating under rough conditions are the current trends in helicopter design. A big part of this is to improve tracking performance and disturbance rejection. Control design of autonomous flying systems has become a very challenging area of research.

Up to some extent, the effects of the wind in an aerial environment are similar to the effects of sea currents in a maritime environment, but the variable nature of the wind brings a degree of complexity that makes it relevant for a separate study. According to the current state of the art, we will base our model of the aerial environment in the simulation of wind and wind gusts, as these seem the most relevant challenges.

It is important to note that the aerial environment considered in this study is a simplified 2D model crafted to study wind and wind gusts effects on the swarm behavior. While our solution can be applied to quadcopters or other rotocrafts with a high level of abstraction regarding motion control, it doesn't include the necessary constraints to be applied in planes or other full aerial vehicles where the controller deals with low level motion control. This would require a motion approach other than the two wheel model considered, falling out of the scope of this work. A sketch of the aerial environment is depicted in Fig. 4.3.

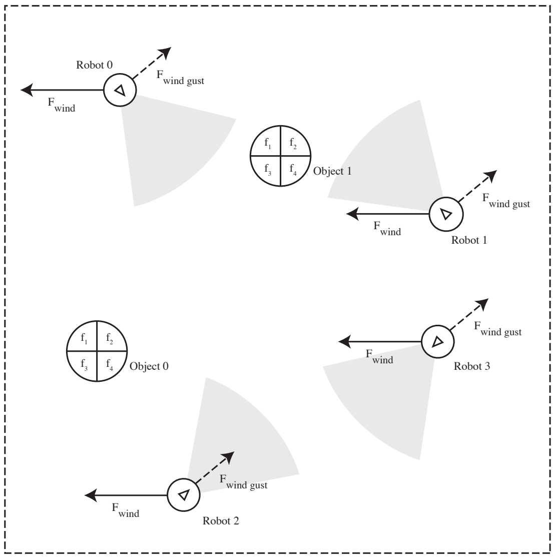

FIGURE 4.3: Schematics of the modeled aerial environment when searching for objects. Object 0 and 1 represent the unidentified objects, with f1 to f4 representing it's features; robot 0 to robot 3 represent the swarm; grey filled sensors represent the local features sensor of each robot; F_{wind} represents the random constant dragging current applied to the robots' movement with magnitude [-2,2] cm/s; $\mathrm{F}_{\text{wind gust}}$ represents the random constant dragging current applied to the robots' movement with magnitude [-0.1,0.1] cm;

Our model of an aerial environment is based on (i) constant wind (F_{wind}) and (ii) random wind gusts ($F_{\text{wind gust}}$). The constant wind is modeled like the constant sea current described in section 4.1.2: a vertical and an horizontal magnitude between [-2,2] cm/s for each axis. Wind gusts have a period between [0,20] seconds sorted at the beginning of each wind gust, together with the magnitude, being constant throughout the entire period. At the end of each wind gust period, it is randomly sorted whether the next period will be silent or windy. Constant and random wind magnitude, gust duration and whether a gust is present or not are all random values drawn from a uniform distribution.

Throughout the next sections, we will test the controllers resulting from four different types of evolution: (i) environment specific, (ii) ideal, (iii) noisy and (iv) hybrid in each of the different environments described above.

4.2 Evolving and testing the solution on different setups

Machine learning mechanisms like the search algorithms used to obtain the controllers, learn from observing the environment and measuring the performance of each candidate solution. As we place the robots in different settings, the optimization will follow different paths and we obtain different solutions, specifically optimized to the setup the evolutionary process was conducted within. In this section, we conduct the evolutionary process in four main setup categories: (i) in each environment, (ii) in an ideal setup, (iii) in a noisy environment and (iv) in a hybrid scenario. The hybrid scenario consists in each sample being conducted in a different environment (terrain, marine or aerial). We will then transfer the controllers obtained to each environment and test its performance.

All the experiments in this section will be carried out 10 times (evolutionary runs) during 2000 generations with a population of 100 individuals tested during 1000 timesteps, each experiment repeated 30 times (samples). A total of 60 evolutionary processes were conducted to present the results in this chapter, taking 29 days to complete on a computer grid with average availability of 75 workers, with a maximum of 3 evolutionary processes evolving in parallel.

4.2.1 Environment-specific Evolution and Results

To set a benchmark for the target controller behavior in each environment, we conducted separate evolutionary processes in the terrain, marine and aerial environments. This way, we will obtain controllers specifically optimized for each environment. If we find controllers obtained via other methods to perform comparably good as the controllers obtained in this section, the generic solution will be validated.

Controllers will be tested not only in the environment they were evolved in but also in all the others. The evolutionary process was conducted to optimize the fitness function set in Eq. 3.4 with the configuration detailed in Table 3.1 in the environments described in section 4.1.

As in Chapter 3, the tests are done to the best controller resulting from the evolution, with an average of 100 samples during 10000 time steps.

4.2.1.1 Terrain Evolution

When evolving the controller in the terrain environment, the best controller resulting from the evolutionary process scored: (i) an average fitness of 2874 ± 1922, (ii) an "enemy identification rate" of 54%, (iii) an "enemy catch rate" of 42% and (iiv) a "friends and unidentified objects catch rate" of 1% and 5%, respectively. Results obtained when tested on an ideal environment. Controller performance in the different scenarios is condensed on Table 4.1 and Fig. 4.4.

TABLE 4.1: Terrain environment evolution: tested in multiple environments

Scenario	Fitness ± Stdev	Enemies identified (%)	Enemies caught (%)	Friends caught (%)	Unidentified caught (%)
Ideal Environment	2874 ± 1922	54%	42%	1%	5%
Noisy Environment	-529 ± 1713	5%	0%	0%	4%
Terrain Environment	2158 ± 1433	41%	32%	1%	3%
Marine Environment	694 ± 1415	25%	11%	0%	3%
Aerial Environment	752 ± 801	22%	13%	0%	3%
Hybrid Environment	1263 ± 1056	31%	20%	1%	3%

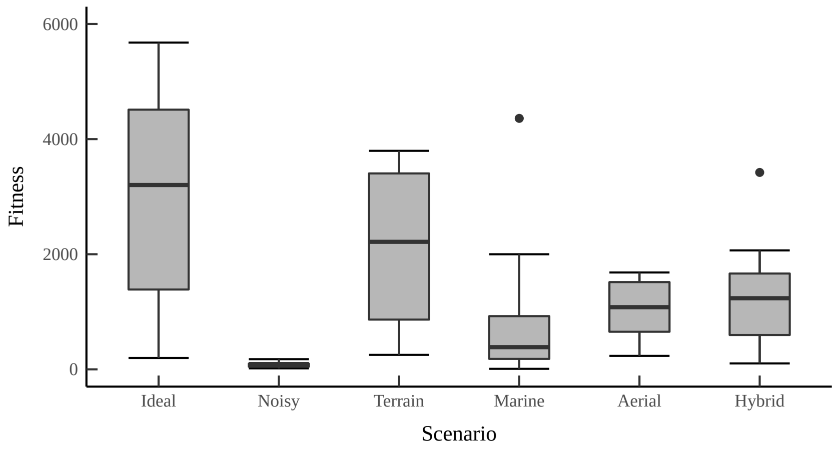

FIGURE 4.4: Terrain environment evolution: tested in multiple environments

In this scenario, 80% of the evolutionary runs successfully evolved a controller that solved the proposed task. When observing the solutions, two distinct behaviors were found. 40% evolved a behavior where the robots disperse and search around the environment until an object gets in sight. In another 40%, the evolution went on a different direction: a behavior in which the robots follow each other around the arena doing a group search. In test cases where the random obstacles confined rooms inside the arena, groups of robots spreaded and patrolled only one room with circular motions around themselves. 20% of the evolutionary runs evolved a backwards motion behavior and were not able to solve the problem. On average, a good behavior was found around generation 800.

We notice the controller was not able to adapt to the noisy environment and scored a low performance on the marine and aerial environments.

4.2.1.2 Marine Evolution

When evolving the controller in the marine environment, the best controller resulting from the evolution scored: (i) an average fitness of 3067 $\pm$ 1668, (ii) an "enemy identification rate" of 56%, (iii) an "enemy catch rate" of 46%, (iv) a "friends and unidentified object catch rate" of 2% and 7%, respectively. Results when tested on an ideal environment. Controller performance in the different scenarios is condensed on Table 4.2 and Fig. 4.5.

TABLE 4.2: Marine environment evolution: tested in multiple environments

Scenario	Fitness ± Stdev	Enemies identified (%)	Enemies caught (%)	Friends caught (%)	Unidentified caught (%)
Ideal Environment	3067 ± 1668	56%	46%	2%	7%
Noisy Environment	-492 ± 1600	4%	0%	0%	4%
Terrain Environment	2191 ± 1261	43%	34%	2%	5%
Marine Environment	3473 ± 1549	59%	51%	2%	7%
Aerial Environment	737 ± 684	24%	13%	0%	4%
Hybrid Environment	2242 ± 1120	42%	34%	1%	5%

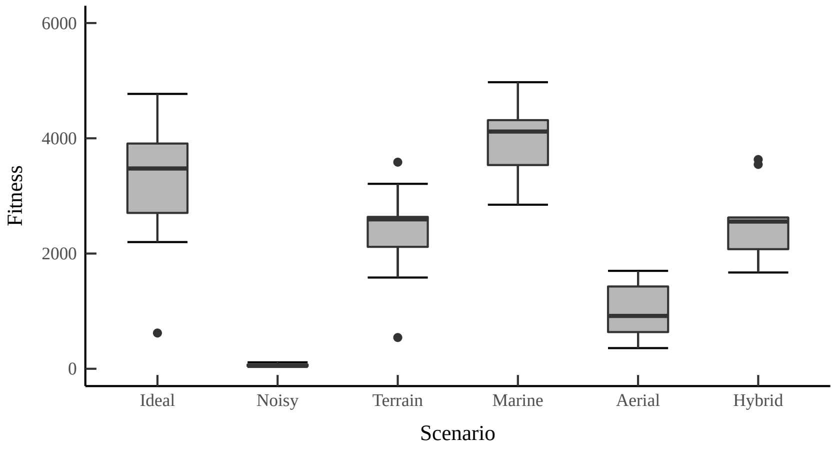

FIGURE 4.5: Marine environment evolution: tested in multiple environments

In this scenario, 90% of the evolutionary runs evolved a controller capable of solving the proposed task. 10% evolved a backwards motion behavior and were unable to solve the task. 70% evolved the dispersed search behavior, with the remaining 20% evolving the teammates follow group search behavior. On average, a good result to solve the task was found generation 800, with fitness stabilizing since then.

We notice the controller was not able to adapt to the noisy environment and scored a low performance on the aerial environment.

4.2.1.3 Aerial Evolution

When evolving the controller in the aerial environment, the best controller resulting from the evolution scored: (i) an average fitness of 2385 ± 1026, (ii) an "enemy identification rate" of 48%, (ii) an "enemy catch rate" of 40%, (iii) a "friends and unidentified object catch rate" of 6% and 7%, respectively. Results when tested on an ideal environment. Controller performance in the different scenarios is condensed on Table 4.3 and Fig. 4.6.

Table 4.3: Aerial environment evolution: tested in multiple environments

Scenario	Fitness ± Stdev	Enemies identified (%)	Enemies caught (%)	Friends caught (%)	Unidentified caught (%)
Ideal Environment	2385 ± 1026	48%	40%	6%	7%
Noisy Environment	92 ± 210	15%	2%	3%	1%
Terrain Environment	1715 ± 799	37%	31%	5%	5768%
Marine Environment	768 ± 901	26%	12%	1%	3%
Aerial Environment	1428 ± 618	35%	23%	2%	4%
Hybrid Environment	1334 ± 777	34%	23%	3%	4%

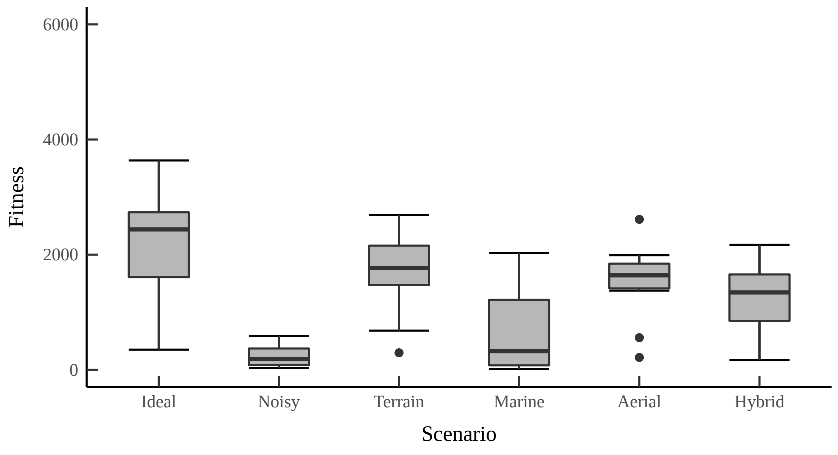

Figure 4.6: Aerial environment evolution: tested in multiple environments

In this scenario, 90% of the evolutionary runs were able to solve our initial task. 10% was not able to solve the task and, evolving a backwards motion behavior. In all controllers, the robots were always close together: search dispersion found in other controllers was not found here. For this controller, performance was globally lower than the controllers on previous sections, except on the aerial environment. On average, a good behavior was found around generation 800, with fitness stabilizing ever since.

We notice the controller was not able to adapt to the noisy environment and scored a low performance on the marine environment.

4.2.2 Ideal Evolution applied to Multiple Environments

The results for the controller evolved under ideal conditions is explained in section 3.4.2. In this section, the controller will be transfered to each environment, with the respective performance condensed on Table 4.4 and Fig. 4.7.

TABLE 4.4: Ideal environment evolution: tested in multiple environments

Scenario	Fitness ± Stdev	Enemies identified (%)	Enemies caught (%)	Friends caught (%)	Unidentified caught (%)
Ideal Environment	4239 ± 2573	71%	64%	2%	7%
Noisy Environment	-90 ± 515	6%	0%	0%	1%
Terrain Environment	2576 ± 1340	45%	37%	1%	3%
Marine Environment	843 ± 1518	25%	12%	0%	2%
Aerial Environment	836 ± 748	22%	13%	0%	3%
Hybrid Environment	1557 ± 1112	33%	23%	0%	3%

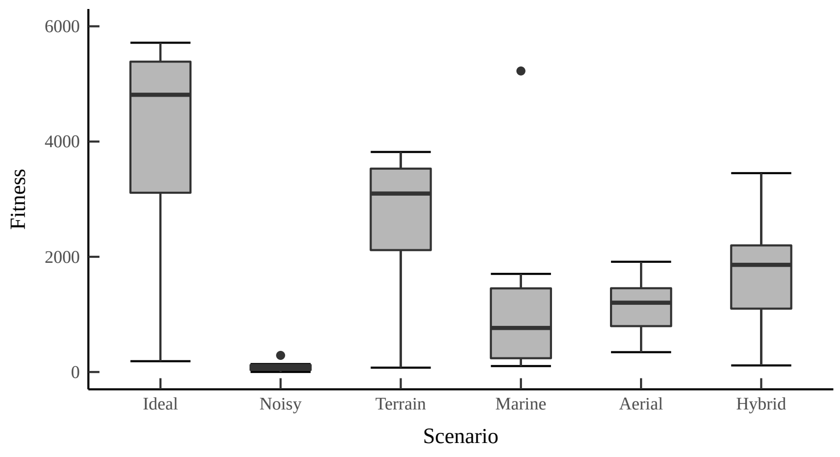

FIGURE 4.7: Ideal environment evolution: tested in multiple environments

When transferring the controller to the different environments, we notice the controller was not able to adapt to the noisy environment and scored a low performance on the marine and aerial environments. Due to it's characteristics, these seem to be the hardest for the ideal evolved controller to adapt to. The terrain environment appears to be the easiest for the ideal evolved controller to deal with.

4.2.3 Noisy Evolution and Results

Introducing noise on the ANN Inputs during the evolutionary process is one of the known ways of creating a solution that is able to cope with slightly different conditions than the ideal environments usually used during training, thus boosting the ability to cross the reality gap. In other words, noise can be seen as an abstract and multi-purpose way of generating a more robust solution.

In a separate study, we introduced the concept of Genome Variations (GV) [32]. This study proposes a technique that is based on mutating the individual controllers of each robot prior to evaluation. Variations are obtained by adding noise to the weights of the ANN controlling the robots. GV represents a novel approach to the evolution of robust behaviors. Although this study revealed some interesting insights, the approach is not as viable as the classic input noise methods also studied in the article. As such, the latter will be used in this section.

All sensors are affected by the noise, with a fixed offset of [-0.1,0.1] and random noise [-0.1,0.1] for each reading, as suggested in [32]. For object features, a 10% probability of having each binary reading state reversed is used, a value equivalent to the previous. Offset, noise values and binary state reversions are random processes drawn from a uniform distribution

When evolving the controller in the noisy environment, the best controller resulting from the evolution scored: (i) an average fitness of 2120 ± 428, (ii) an "enemy identification rate" of 59%, (ii) an "enemy catch rate" of 47%, (iii) a "friends and unidentified objects catch rate" of 42% and 4%, respectively. Results when tested on an ideal environment. Controller performance in the different scenarios is condensed on Table 4.5 and Fig. 4.8.

Table 4.5: Noisy environment evolution: tested in multiple environments

Scenario	Fitness ± Stdev	Enemies identified (%)	Enemies caught (%)	Friends caught (%)	Unidentified caught (%)
Ideal Environment	2120 ± 428	59%	47%	42%	4%
Noisy Environment	2186 ± 319	58%	48%	43%	4%
Terrain Environment	1526 ± 209	45%	36%	32%	4%
Marine Environment	191 ± 714	25%	15%	13%	6%
Aerial Environment	443 ± 193	22%	10%	9%	2%
Hybrid Environment	809 ± 388	31%	20%	18%	3%

The noisy evolution is not only the lowest performing solution of all, but the

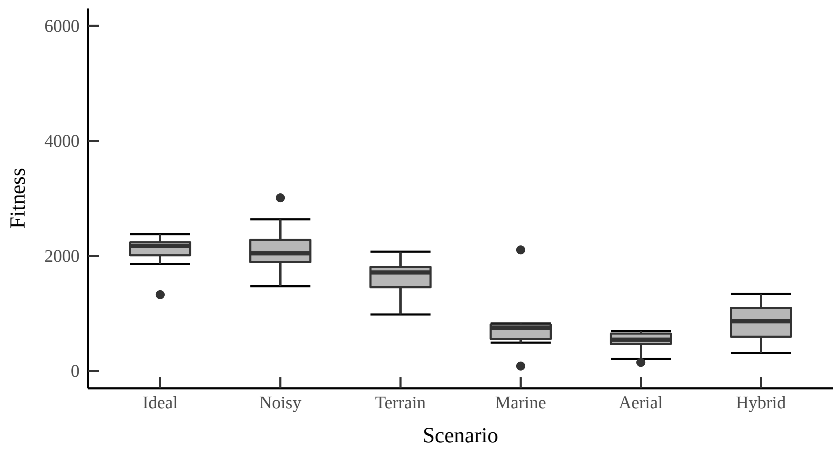

FIGURE 4.8: Noisy environment evolution: tested in multiple environments

first that globally failed in solving the proposed task. When the behavior is observed, we notice that in 100% of the evolutionary runs, the search and identification behavior appears correct and similar, but the swarm catches both enemies and friends, not being able to distinguish them. The controller thus failed mainly in the feature's categorization and object identification. In 90% of the evolutions, a shaky, rippled or less smooth motion with more aggressive and quantized movements was observed. Fitness stabilized around generation 1000.

4.2.4 Hybrid Evolution and Results

Another possible method capable of generating a more robust and scalable controller is by integrating all the environments in the same evolutionary process, thus generating a controller that is optimized to all the different environments: a hybrid controller.

In this scenario, the fitness of every individual used for the selection phase will be the average performance of the controller on each of the environments. In each experiment, the solution will be tested on the terrain, marine and aerial environments for $\frac{1}{3}$ of the total number of samples each. This will conduct to a solution that, in theory, is equally optimized for all 3 environments.

When evolving the controller in the hybrid scenario, the best controller resulting from the evolutionary process scored: (i) an average fitness of 3031 ± 1292, (ii) an "enemy identification rate" of 54%, (iii) an "enemy catch rate" of 45% and (iv) a "friends and unidentified objects catch rate" of 4% and 6%, respectively. Results when tested on an ideal environment. Controller performance in the different scenarios is condensed on Table 4.6 and Fig. 4.9.

Table 4.6: Hybrid environment evolution: tested in multiple environments

Scenario	Fitness ± Stdev	Enemies identified (%)	Enemies caught (%)	Friends caught (%)	Unidentified caught (%)
Ideal Environment	3031 ± 1292	54%	45%	4%	6%
Noisy Environment	157 ± 220	16%	0%	0%	0%
Terrain Environment	2108 ± 951	41%	34%	3%	4%
Marine Environment	2553 ± 1176	48%	38%	2%	6%
Aerial Environment	1057 ± 483	27%	16%	1%	3%
Hybrid Environment	2004 ± 881	40%	31%	2%	4%

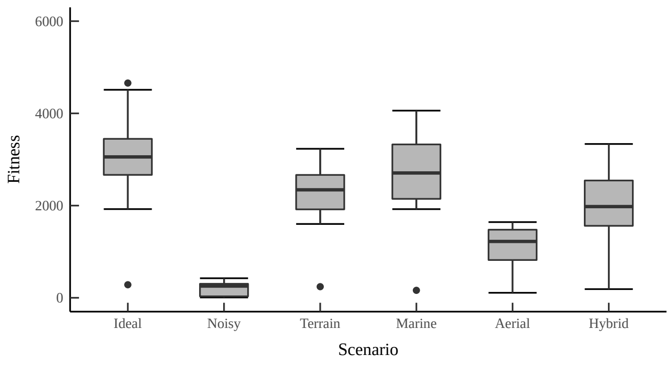

Figure 4.9: Hybrid environment evolution: tested in multiple environments

In this scenario, 90% of the evolutionary runs generated a controller that is able to solve the proposed task. 40% evolved the follow teammates search behavior and 40% evolved the regular dispersed search behavior. 10% evolved a backwards motion that was able to solve the tasks, and 10% evolved a backwards motion behavior that was unable to solve the task. On average, a good behavior was found around generation 500.

We notice the controller was not able to adapt to the noisy environment, and scored the lowest performance in the aerial environment. Results for the hybrid controller on each environment reveal to be on pair with the environment-specific evolution described in section 4.2.1. This will be further analyzed in the following section.

4.3 Discussion and Comparison

After reviewing the results of each evolution scenario we can extract four main insights: (i) environment specific evolution although strong was not always the best option for the respective environment: the terrain evolved controller obtained a lower performance on the terrain environment than the ideal and marine controllers; (ii) contrary to our initial hypothesis, environment noise was not able to generate a robust controller, as the solution was not able to cope with this type and magnitude of environment noise; (iii) hybrid evolution gave us results on par with environment-specific evolved controllers, validating this approach.

We will now turn our attention to the results presented in section 4.2 in another way. Fig. 4.10, shows the results for each of the controllers, grouped by environment they were tested on.

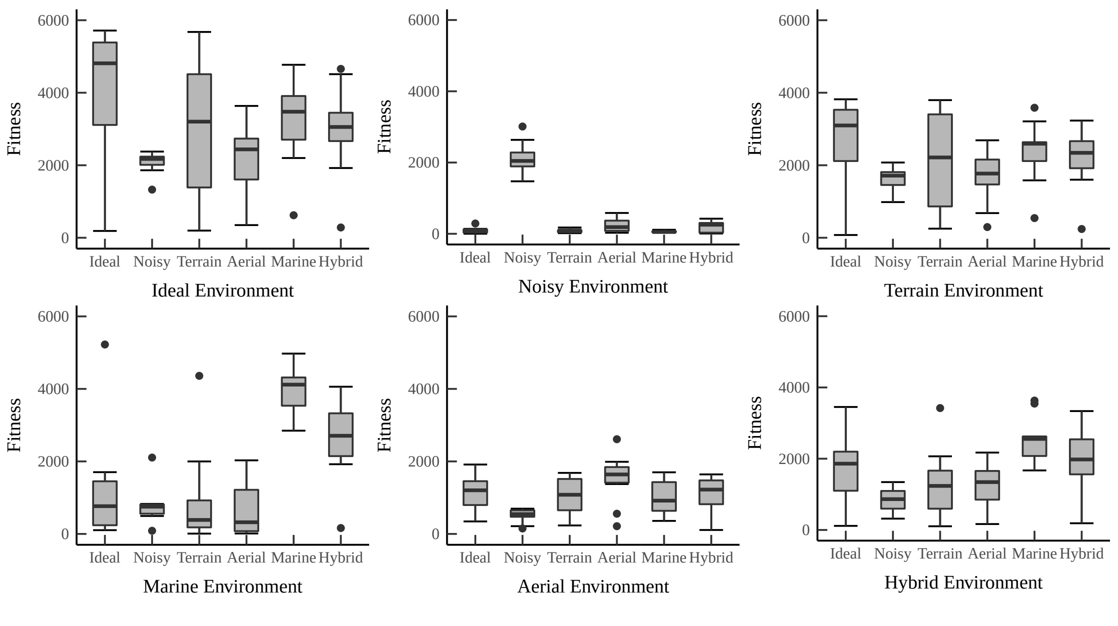

FIGURE 4.10: Final comparison: all controllers tested in all scenarios

The terrain environment is the example of a test case where the environment specific evolution was not the best option. The ideal and marine evolved controllers had better performance on the terrain environment, with average fitness of 2576 ± 1340 and 2191 ± 1261, respectively, while the terrain evolved controller scored a fitness of 2158 ± 1433. Although the margin is small enough to state that all three controllers are equally capable of solving the task, it stands out the fact that the terrain evolved scenario was not the best fit to solve the task in the environment it was trained in, with the ideal evolved behavior presenting a global average 15% fitness increase. This seem to happen due to the complexity of the scenario: the obstacles that increase the difficulty of searching and identifying the objects to simulate a real-world terrain environment also prevent the evolution from extracting the global patterns of object identification and catching as well as it did on the ideal environment. It is noteworthy how a more generic and ideal scenario was able to generate a behavior more adaptable to a constrained environment than the solution evolved within it. Also, the characteristics of the terrain environment conducted the evolution to a behavior in which the swarm separates in small search groups strategically placed in spaces confined by the obstacles.

When tested on the noisy environment, all controllers failed to solve the task. Although the noise magnitude used in these experiments gave us good results in previous studies [32], it appears to be destructive for this task. On the previous studies, we used a simple aggregation and formation task with identical noise applied, with good results to create a more robust behavior. The controller we present in this work shares many of the same sensors and actuators as the solution on the previous study (robot sensors, wall sensors, two-wheels actuator), making this result counterintuitive. On the other hand, the biggest difference between the two architectures is the shared features sensor. While the search and identification portion of the behavior seems correct, the categorization was the main variable to fail in the controller (that caught both enemies and friends), leading us to conclude the shared features sensor was the bottleneck that caused the noisy evolved controller to fail, being the component less prone to noise.

The marine environment is the environment with the biggest discrepancy between the environment specific evolution performance and the remaining, with the environment specific controller scoring an average fitness of 3473 $\pm$ 1549. Hybrid evolved controller on this environment scored a lower fitness of 2242 $\pm$ 1120. The ideal, terrain and aerial evolved controller scored the lowest fitness by a big margin:

843 ± 1518, 694 ± 1415 and 768 ± 901, respectively. The robot movement inertia is the main difference in this environment. This results shows us that although the adaption to this characteristic is needed (low fitness on the ideal evolution), the adaption is not hard for the evolution to handle (high fitness in the environment specific and hybrid controllers). Direct observation of the behavior presents no visible differences to the remaining solutions.

The aerial environment presented the lowest global fitness values among the three environments. With no clear performance distinction from the environment specific solution, we conclude that the evolution was not able generate a controller that compensates for the wind gusts. Observing the behavior, we notice that when the wind gusts appear, the robots lose control of the object being identified. Controllers evolved in the aerial environment revealed a tendency to always keep close together (behavior found on 90% of the evolutionary runs). This tendency was not observed in the remaining scenarios and represents a specific path the aerial evolution followed, possibly keeping teammates close to use them as a reference to acquire spacial awareness when the wind gusts drag the robots out of their position.

Direct observation of all behaviors revealed three distinct patterns that the evolution followed: (i) in about 10% of the evolutionary runs, a backwards motion behavior was evolved, leading that evolutionary run to not be able to bootstrap the evolution and presenting a global inability to solve the task; (ii) around 15% of the controllers evolved a follow teammates behavior with object search done in a group with robots following each other(behavior mostly found on the terrain and hybrid controllers); (iii) around 35% of controllers evolved a behavior in which the robots disperse to search and then aggregate to identify and catch.

We noted that on all previous cases, the hybrid controller performance revealed to be on par with the environment specific results in terms of fitness. To further analyze these results, the differences between the environment specific controllers and the hybrid controller are condensed on Table 4.7, for: (i) "enemies identified ratio", (ii) "enemies caught ratio" and (iii) "friends and unidentified objects caught ratio".

Table 4.7: Environment specific controllers compared to the hybrid evolved controller in each scenario

Environment	Terrain Environment		Marine Environment		Aerial Environment	
Controller	Env. specific	Hybrid (± diff)	Env. specific	Hybrid (± diff)	Env. specific	Hybrid (± diff)
Enemies identified (%)	41%	41% (0%)	43%	48% (+5%)	35%	27% (-8%)
Enemies caught (%)	32%	34% (+2%)	36%	38% (+2%)	23%	16% (-7%)
Friends caught (%)	1%	3% (+2%)	2%	2% (0%)	2%	1% (-1%)
Unidentified caught (%)	3%	4% (+1%)	5%	6% (+1%)	4%	3% (-1%)

We notice that the differences between the two approaches range from a positive performance of [0,5] percentage points for the hybrid controller in the terrain and marine environment and a slight degradation of performance of [1,8] percentage points in the aerial environment.

We conclude that the hybrid controller reveals to be equivalent to the environment specific controllers in the terrain and marine environments, and worse on the aerial environment. Still, the differences found between these are of small magnitude. In terms of observable behavior, there are no visible differences as both solve the task in the same manner. We can state that the performance for the hybrid controller on the terrain, marine and aerial environments is on pair with evolution specific controllers, differing only by a small negligible margin with no clear performance impact.

Chapter 5

Conclusions

In this dissertation, we proposed a novel approach for swarm robotics environment perception. This approach is different from the remaining state of the art for two main reasons: (i) the controller is obtained using EAs and (ii) the study is focused on scaling the approach to multiple environments.

We conducted the study in a simulation scenario, starting with the development of the behavior model in ideal conditions with unidentified objects appearing from left to right on the arena. We obtained: (i) an "enemy identification rate" of 77%, an "enemy catch rate" of 73% and a "friends and unidentified objects catch rate" of 2% and 8% respectively (false positives).

In the next step, we increased the complexity of the problem. The unidentified objects can now appear from any side of the screen moving to the opposite side, with the possibility of having two objects on screen at the same time. This controller scored: (i) an "enemy identification rate" of 71%, an "enemy catch rate" of 64% and a "friends and unidentified objects catch rate" of 2% and 7%, respectively. The evolved behavior consists in performing a dispersed search around the arena, getting closer to the objects when an enemy feature is detected. When robots gather around the object, one of them catches it. Attention given to friend features is lower, so robots didn't gather around the friend objects most times, nor caught them.

In chapter 4, we focused our attention in scaling the previously obtained controller to multiple environments. We modeled a simulation environment for: (i) a terrain environment based on obstacles randomly placed around the environment, (ii) a marine environment with constant currents and inertia in the robots' movements and (iii) a aerial environment with a constant current and wind gusts. Also, we selected 2 main scenarios that are known to evolve more robust behaviors: (i) noisy evolution and (ii) a hybrid evolution in the multiple scenarios. These were compared to the ideal evolution scenario.

When observing the evolved behaviors, two main categories can be extracted: in the first, the robots evolved a behavior in which the team performs a dispersed search around the arena and then aggregate around the object to proceed with the identification; in the second, the robots follow each other in circular paths around the environment once again aggregating towards the object to identify. The identification process followed very similar behavior in all experiments: circumnavigating the object while front-facing it until the identification is complete. Specialization was also observed on the environment-specific evolutions: in the terrain evolved controller the swarm had a tendency to separate in groups and search inside the areas confined by the obstacles.

The noisy evolution not only failed to evolve a more robust and scalable solution, but failed to solve the task at all, as the noise magnitude revealed to be destructive for this task. While search and identification appeared similar and correct, identification failed as the swarm caught all objects instead of only enemies. Behavior observed here appeared less smooth and more quantized: when a controller is evolved in a noisy environment, the evolution follows a path that benefits the controllers that are more robust, more difficult to severally impact performance with slight differences in the input values. This explains the quantized behavior observed. Globally, noisy evolved controllers tend to have clear and fixed values for its outputs (in this case the wheels), while non-noisy evolved solutions converge to more smooth curves of motion of the output variables.

The global objective of this work was to test and compare several ways of developing a controller capable of collectively identifying a set of objects and act upon multiple types of environments based on a categorization of the objects identified. This objective was successfully completed as we demonstrated how EAs could synthesize a controller capable of solving this task. Additionally, we demonstrated the flexibility of our machine learning approach to generate a controller that could achieve good results in multiple environments by evolving a solution globally optimized for them: the hybrid solution. Although environment-specific controllers globally outperformed the hybrid controller in the respective environment, the difference between the two are small enough to state that both controllers are equally capable of solving our task, with the advantage of the hybrid controller presenting a more global solution.

Of all environments, the biggest difficulty for the controller appeared to be on the aerial environment, specifically the the wind gusts, that the controller had difficulty in compensating. Future work on this area could reside in optimizing this controller for better results in the different environments. For example, giving the controller access to a sensor that detects wind gusts could help the robot compensate them and boost the performance on the aerial environment.

Bibliography

[1] P. M. Fitzpatrick, "Perception and perspective in robotics," *Proceedings of the 25th Annual Conference of the Cognitive Science Society*, 2003.

[2] E. Mascalzoni and L. Regolin, "Animal visual perception," *Wiley Interdisciplinary Reviews: Cognitive Science*, vol. 2, no. 1, pp. 106–116, 2011.

[3] M. A. Lewis, A. H. Fagg, and A. Solidum, "Genetic Programming Approach to the Construction of a Neural Network for Control of a Walking Robot," *In IEEE International Conference on Robotics and Automation*, pp. 2618–2623, 1992.

[4] D. Cliff, P. Husbands, and I. Harvey, "Evolving visually guided robots," *Proceedings of the Second International Conference on Simulation of Adaptive Behavior (SAB)*, no. July 1992, pp. 374–383, 1993.

[5] K. Jim, C. L. Giles, and B. G. Horne, "Effects of Noise on Convergence and Generalization in Recurrent Networks," *Advances in Neural Information Processing Systems (NIPS) 7*, p. 649, 1995.

[6] L. Merino, F. Caballero, J. R. Martínez-de Dios, J. Ferruz, and A. Ollero, "A cooperative perception system for multiple UAVs: Application to automatic detection of forest fires," *Journal of Field Robotics*, vol. 23, no. 3-4, pp. 165–184, 2006.

[7] M. T. J. Spaan, "Cooperative Active Perception using POMDPs," *October*, pp. 4800–4805, 2010.

[8] M. T. J. Spaan, T. S. Veiga, and P. U. Lima, "Active cooperative perception in network robot systems using POMDPs," *IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings*, pp. 4800–4805, 2010.

[9] Q. V. Le, A. Saxena, and A. Y. Ng, "Active Perception : Interactive Manipulation for Improving Object Detection," 2010.

[10] M. T. J. Spaan, T. S. Veiga, and P. U. Lima, "Decision-theoretic planning under uncertainty with information rewards for active cooperative perception," *Autonomous Agents and Multi-Agent Systems*, vol. 29, no. 6, pp. 1157–1185, 2014.

[11] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[12] S. M. Prabhu and D. P. Garg, "Artificial Neural Network Based Robot Control: An Overview," *Journal of Intelligent and Robotic Systems*, vol. 15, no. 1993, pp. 333–365, 1996.

[13] J. Chen and S. Wermter, "Continuous Time Recurrent Neural Networks for Grammatical Induction," *International Conference on Artificial Neural Networks,1998*, pp. 381–386, 1998.

[14] Stefano Nl, "Institute of Psychology CNR-Rome," *Population (English Edition)*, vol. 1997, no. May, pp. 1–27, 1994.

[15] N. Jakobi, P. Husbands, and I. Harvey, "Noise and the Reality Gab: The Use of Simulation in Evolutionary Robotics," *Lecture Notes in Computer Science*, vol. 929, pp. 704–720, 1995.

[16] D. Floreano and J. Urzelai, "Evolutionary Robots: The Next Generation," *The 7th International Symposium on Evolutionary Robotics (ER2000): From Intelligent Robots to Artificial Life*, pp. 231–266, 2000.

[17] C. Hartland and N. Bredèche, “Evolutionary robotics, anticipation and the reality gap,” *2006 IEEE International Conference on Robotics and Biomimetics, ROBIO 2006*, pp. 1640–1645, 2006.

[18] M. Eaton, “Evolutionary humanoid robotics: past, present and future,” *Lecture Notes in Computer Science*, vol. 4850, p. 42, 2007.

[19] M. Duarte, S. Oliveira, and A. L. Christensen, “Hierarchical evolution of robotic controllers for complex tasks,” *2012 IEEE International Conference on Development and Learning and Epigenetic Robotics, ICDL 2012*, no. June, 2012.

[20] A. Iscen, “Learning Tensegrity Locomotion Using Open-Loop Control Signals and Coevolutionary Algorithms Abstract,” *Artificial Life*, vol. 19, no. 3/4, pp. 119–140, 2013.

[21] A. Ahmad, T. Nascimento, A. G. S. Conceicao, A. P. Moreira, and P. Lima, “Perception-driven multi-robot formation control,” *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 1851–1856, 2013.

[22] A. Rauch, S. Maier, F. Klanner, and K. Dietmayer, “Inter-vehicle object association for cooperative perception systems,” *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, no. Itsc, pp. 893–898, 2013.

[23] S.-W. Kim, B. Qin, Z. J. Chong, X. Shen, W. Liu, M. H. Ang, E. Frazzoli, and D. Rus, “Multivehicle Cooperative Driving Using Cooperative Perception: Design and Experimental Validation,” 2015.

[24] T. Rodrigues, M. Duarte, M. Figueiró, V. Costa, S. M. Oliveira, and A. L. Christensen, “Overcoming limited onboard sensing in swarm robotics through local communication,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9420, pp. 201–223, 2015.

[25] A. Dias, J. Capitan, L. Merino, J. Almeida, P. Lima, and E. Silva, “Decentralized target tracking based on multi-robot cooperative triangulation,”

2015 IEEE International Conference on Robotics and Automation (ICRA), pp. 3449–3455, 2015.

[26] B. Burchfiel and G. Konidaris, "Generalized 3D Object Representation using Bayesian Eigenobjects,"

[27] M. Duarte, F. Silva, T. Rodrigues, S. M. Oliveira, and A. L. Christensen, "{JBotEvolver}: A versatile simulation platform for evolutionary robotics," *Proceedings of the International Conference on the Synthesis & Simulation of Living Systems (ALIFE)*, pp. 210–211, 2014.

[28] M. Duarte, V. Costa, J. Gomes, T. Rodrigues, F. Silva, S. M. Oliveira, and A. L. Christensen, "Evolution of collective behaviors for a real swarm of aquatic surface robots," *PLoS ONE*, vol. 11, no. 3, pp. 1–25, 2016.

[29] J. Pflimlin, P. Soueres, and T. Hamel, "Hovering flight stabilization in wind gusts for ducted fan UAV," *2004 43rd IEEE Conference on Decision and Control (CDC) (IEEE Cat. No.04CH37601)*, vol. 4, no. January 2005, pp. 3491–3496, 2004.

[30] T. Cheviron, F. Plestan, and A. Chriette, "A robust guidance and control scheme of an autonomous scale helicopter in presence of wind gusts," *International Journal of Control*, vol. 82, no. 12, pp. 2206–2220, 2009.

[31] F. Leonard, A. Martini, and G. Abba, "Robust nonlinear controls of model-scale helicopters under lateral and vertical wind gusts," *IEEE Transactions on Control Systems Technology*, vol. 20, no. 1, pp. 154–163, 2012.

[32] P. Romano, L. Nunes, A. L. Christensen, M. Duarte, and S. M. Oliveira, "Genome Variations," in *Robot 2015: Second Iberian Robotics Conference: Advances in Robotics, Volume 1* (L. P. Reis, A. P. Moreira, P. U. Lima, L. Montano, and V. Muñoz-Martinez, eds.), pp. 309–319, Cham: Springer International Publishing, 2016.