Hybrid Control for Large Swarms of Aquatic Drones

Miguel Duarte, Sancho Moura Oliveira and Anders Lyhne Christensen

Instituto de Telecomunicações &
Instituto Universitário de Lisboa (ISCTE-IUL), Lisboa, Portugal
{miguel_duarte,sancho.oliveira,anders.christensen}@iscte.pt

Abstract

Maritime tasks, such as surveillance and patrolling, aquaculture inspection, and wildlife monitoring, typically require large operational crews and expensive equipment. Only recently have unmanned vehicles started to be used for such missions. These vehicles, however, tend to be expensive and have limited coverage, which prevents large-scale deployment. In this paper, we propose a scalable robotics system based on swarms of small and inexpensive aquatic drones. We take advantage of bio-inspired artificial evolution techniques in order to synthesize scalable and robust collective behaviors for the drones. The behaviors are then combined hierarchically with preprogrammed control in an engineered-centric approach, allowing the overall behavior for a particular mission to be quickly configured and tested in simulation before the aquatic drones are deployed. We demonstrate the scalability of our hybrid approach by successfully deploying up to 1,000 simulated drones to patrol a 20 km long strip for 24 hours.

Introduction

Coastal countries have faced an increased spending over the years in order to carry out maritime tasks. In Italy, for instance, the problem of illegal immigration (Monzini, 2007; Carling, 2007) and organized crime (Lutterbeck, 2006) has contributed for the growth of the Guardia di Finanza’s budget from $1.10B to $3.21B, during the 1990s, with an increase both in personnel (28%) and equipment, which, in 1999, counted 582 boats (78% increase), 96 helicopters (41% increase), and a total of 14 airplanes (Lutterbeck, 2004). In Spain, immigrants crossing the Gibraltar Strait through Morocco have lead the Spanish government to implement the Sistema Integrado de Vigilancia Exterior (SIVE), which is composed of military technology such as fixed and mobile radars, infrared sensors, and traditional aquatic and aerial vehicles (Lutterbeck, 2006).

Since current state-of-the-art systems require a large human crew in order to successfully execute maritime tasks, efforts have been made to adapt unmanned vehicle technology, such as unmanned aerial vehicles (UAV) and unmanned surface vehicles (USV) for these tasks. However, current UAV and USV systems tend to be expensive to acquire and operate, and therefore tend to be comprised of only a small number of units (Schwing, 2007). Examples are the military-grade General Atomics MQ-9 Reaper UAV, and the Protector USV (Yan et al., 2010). Only recently have technological advances made it possible (and affordable) for researchers to begin to study the use of medium or large-sized swarms of inexpensive and unmanned autonomous vehicles (Manley, 2008).

In this paper, we propose a system composed of a swarm of relatively simple aquatic drones with decentralized control for maritime missions. Our proposed system is composed of potentially hundreds or thousands of small autonomous aquatic drones. Such decentralized systems present numerous potential benefits compared with monolithic robotic systems (Bayindir and Şahin, 2007). On the one hand, the use of multiple drones introduces redundancy in the system, which reduces the impact of hardware failures and has the potential to improve operational efficiency by allowing a larger area to be covered simultaneously. On the other hand, automating such missions considerably reduces the maintenance cost (smaller human crew) while increasing availability and scalability.

We employ our novel hybrid control approach (Duarte et al., 2014b) that combines evolutionary robotics (ER) techniques and preprogrammed logic to facilitate configurable, decentralized control. We evolve several simple self-organized behaviors, which would be challenging to manually program. The use of ER techniques enables the synthesis of robust and scalable controllers for simple tasks, which are then combined using high-level preprogrammed logic for the complete task. In this way, our approach addresses fundamental challenges in the synthesis and use of self-organized control for real-world swarm systems. By dividing control into individual components, (i) it becomes computationally feasible to evolve self-organized control for large-scale systems, (ii) we diminish bootstrapping issues since solutions to a set of simpler tasks are sought instead of solutions to the more complex, global task, and (iii) control for new missions can be composed based on existing behav-
ioral components.

Our study is novel in three respects: (i) we apply evolutionary techniques to the domain of large swarms of aquatic drones, a combination which to the best of our knowledge is unexplored, (ii) we demonstrate our hybrid control approach applied to a multirobot system with up to 1,000 drones, where it has previously been applied only to single-robot systems (Duarte et al., 2014b) and to a small team of three robots (Duarte et al., 2014a), and (iii) we show that combining evolved and preprogrammed control allows system designers to automatically synthesize self-organized behaviors and then compose control for large-scale maritime missions based on these behaviors.

Background

Multirobot systems are well suited to tasks that require distributed sensing and/or action. Furthermore, the degree of robustness and reliability of the system is potentially high given the inherent redundancy at the unit-level, and hardware failures often have only limited impact on performance of the system as a whole (Farinelli et al., 2004). Manual design of decentralized control for large-scale multirobot systems has, however, proven difficult, since a set of microscopic behavioral rules that give rise to the desired macroscopic, self-organized behavior cannot be derived in the general case (Dorigo et al., 2004).

ER has been used as an alternative to manual programming of decentralized robotic control, since it allows for the automatic synthesis of control and for the self-organization of behavior. An evolutionary algorithm (Goldberg, 1989) optimizes a controller, which is copied to every robot in the system. The algorithm fine-tunes the microscopic rules guiding the individual robots based on the resulting macroscopic behavior, and thus removes the need for manual specification of low-level control (Floreano and Keller, 2010). Artificial neural networks (ANN) are widely (but not exclusively) used in ER as robotic controllers. ANNs provide evolution with a relatively smooth search space (Nolfi and Floreano, 2000), are able to represent general and adaptive solutions (Floreano and Mondada, 1994), and are able to tolerate noise, which is inherent to many real-world sensors and actuators (Kam-Chuen et al., 1996).

In the domain of multirobot systems, evolutionary techniques have been used in a variety of contexts, from robotic soccer (Uchibe and Asada, 2006), to collective transport (Groß and Dorigo, 2009), and to establish and maintain data links with swarms of flying robots (Hauert et al., 2009).

Notable examples of projects in the domain of swarms of aquatic robots are CoCoRo (Schmickl et al., 2011), which focuses on synthesis of control for underwater robots, and ASSISIbf (Halloy et al., 2013), which aims to develop communication channels between robots and animals (fish and honeybees, in particular). While these projects rely on bio-inspired control systems, they are only tangentially related with evolved ANN-based control, such as the one presented in this paper. Although the use of ER in aquatic robots has not yet been widely explored, a few studies have approached the subject. Some examples include the evolution of locomotion behaviors (Ijspeert et al., 2007; Meyer et al., 2003), station keeping for an underwater robot (Moore et al., 2013), and a neuroaugmenting approach to the evolution of centralized control for a small team of UAVs (Praczyk, 2014).

Over the years, researchers have identified certain challenges associated with the application of ER. One of the most prevalent challenges concerns bootstrapping the evolutionary process in complex tasks. If controllers for a relatively complex task are sought, evolution may be unable to find a fitness gradient that leads to adequate solutions (Nelson et al., 2009). Another challenge is the use of evolved control in real robotic hardware. Except for a few cases in which evolution is conducted onboard real hardware (see, for instance (Watson et al., 1999)), the evolution of robotic controllers is conducted offline, in simulation, due to the large number of evaluation necessary to obtain capable controllers. Simulation-specific features, which are not present in the real world, may be exploited by evolution. As a consequence, the process of transferring evolved controllers to real robotic hardware, known as crossing the reality gap, typically fails to preserve the level of performance achieved in simulation (Jakobi, 1997).

Unless very simple tasks are considered, it is difficult to foresee which evolutionary setup might be suitable for solving a particular task (Christensen and Dorigo, 2006). Between the controller, fitness function and evolutionary algorithm, many different combinations of settings are possible. It then becomes necessary to run the computationally intensive evolutionary process, often multiple times with different initial conditions, to assess if a particular setup produces useful solutions. This leads to a time-consuming trial-and-error process. While a few studies have been conducted in which evolution was applied in a more engineered-oriented manner, such attempts have, so far, been ad-hoc (see Groß et al. (2006) for an example). Techniques such as incremental evolution (Mouret and Doncieux, 2008) and task-decomposition (Lee, 1999; Whiteson et al., 2005) have been proposed, but such approaches do not address the semi-automatic synthesis of behavior in a systematic way.

Our hybrid approach to the synthesis of behavioral control addresses these issues by applying systematic sub-division of complex tasks into simpler sub-tasks. By evolving behaviors for simpler sub-tasks, the main challenges of ER can be overcome. On the one hand, if a task is too complex for solutions to be evolved, the sub-division still allows us to apply ER techniques to sub-tasks where manually programming solutions is difficult. On the other hand, the sub-division enables the combined use of preprogrammed and evolved control in an engineering-centric approach (Silva et al., 2014), giving more control over the final solutions.
to human designers than with traditional evolutionary techniques. Preprogrammed control can also be used when certain robot-environment interactions are too difficult to accurately model in simulation, as in the case of tasks that require fine sensorimotor coordination (Duarte et al., 2014b), which can prevent controllers from successfully crossing the reality gap. Moreover, manually programmed behaviors can easily be scaled to long-term operation, while evolved controllers are not typically evolved for long-duration tasks carried out in large physical spaces, due to the computational cost associated with evolution of controllers in such conditions.

Proposed Solution

In this section, we describe the hardware model and software platform necessary to realize the robotics system proposed in this paper. The conceptual aquatic drones used in our system are relatively small (length: 1 m), inexpensive (< $1000) and equipped with a number of sensors, such as a camera, GPS, and short-range communication equipment. The aquatic drones are able to move at speeds up to 10 km/h and are equipped with batteries that operate between two and ten hours depending on motor usage. Development and testing of a hardware platform that meets these requirements is being carried out as part of our ongoing work. The local communication system will add the capability to exchange data between drones, such as their GPS location and mission-related information. In this way, communication enables the drones to adjust their behavior based on the actions of other drones, and to coordinate their collective behavior. The capacity to sense nearby robots allows, for instance, drones to maintain coverage of a patrol zone, even when some of the drones experience hardware failure or leave to execute other tasks.

In our approach, behavior primitives are evolved for particular tasks that the drones need to execute during a mission, such as navigation, patrolling, and intruder detection. In this way, a collection of evolved behaviors is built. The individual controllers are then combined hierarchically using behavior arbitrators, which are decision nodes that delegate control to their sub-controllers (Duarte et al., 2014b). The controllers can have multiple hierarchical layers of both evolved and preprogrammed nodes, allowing for detailed control over behavior. By combining evolved behaviors with top-level preprogrammed control, we rely on evolution to find behaviors based on self-organization, while letting the system designer define the conditions that trigger the execution of the different behaviors.

By using such an engineering-centric approach, behaviors can be added or removed based on the mission’s characteristics without the need for additional computational-intensive processes: if a particular mission does not require, for instance, intruder detection, that behavior can be easily removed from the top-level preprogrammed control, and the existing manual rules can be modified and tested in various simulated scenarios before real drones are deployed.

Simulation and Experimental Setup

We use JBotEvolver (Duarte et al., 2014c), an open-source, multirobot simulation platform, and neuroevolution framework, for evolution of behavioral control. Each neural-network-based controller in our experiments was evolved using a simple generational evolutionary algorithm with parameter values that have led to the synthesis of good solutions in previous studies (Duarte et al., 2014b). We use a population of 100 genomes, and the fitness score assigned to each genome is the mean score obtained in 10 simulations with different initial conditions. The five highest scoring genomes are copied directly to the next generation. Another 19 copies of each genome are made and mutation is applied to each gene with a probability of 10%. A Gaussian offset with a mean of 0 and a standard deviation of 1 is applied when a gene undergoes mutation. Ten evolutionary runs were conducted for each behavior, and each run lasted 100 generations. Genomes consist of floating-point alleles that encode the parameters of a continuous-time recurrent neural network with one hidden layer of fully-connected neurons (Beer and Gallagher, 1992). All controllers evolved for the experiments presented in this paper have five hidden neurons.

Each drone was equipped with a number of different sensors: (i) four drone sensors with a range of 200 m, (ii) a waypoint sensor, which indicates the orientation and distance of a selected GPS waypoint, (iii) four patrol boundary sensors with a range of 50 m, (iv) a Boolean boundary sensor which indicates whether the drone is inside the patrol zone or not, and (v) an intruder sensor, which detects intruders up to a range of 100 m in front of the drone using the camera. All sensors have an opening angle of 90°, and each sensor feeds the ANN with a value in the interval [0, 1], depending on the distance of the closest object inside its field-of-view. Since the camera does not collect depth data, it feeds the ANN with (i) an input that indicates whether an intruder is being detected or not, and (ii) an input that indicates the relative direction to the intruder, in case an intruder is detected.

The drone sensors are implemented by combining local communication with GPS position, while the boundary sensors are implemented based on GPS position and a set of GPS coordinates. The four sensors of each type provide drones with omnidirectional drone sensing and omnidirectional boundary sensing up to the sensors’ respective ranges. The ANN controller’s output layer has three output neurons: two outputs control the speed of both motors, while a third stop motors output turns off the motors if its activation is over a threshold of 0.5.

Experiments and Results

To test the proposed approach, we chose a maritime patrolling and intruder detection task where a swarm of aquatic
Figure 1: Map of the island of Lampedusa in the Mediterranean Sea, with a 20 km by 0.5 km patrol zone. Drones are deployed from two base stations to a random location inside the patrol zone.

![Map of Lampedusa](image)

Figure 2: Representation of the hierarchical controller, with one preprogrammed behavior arbitrator and three evolved behavior primitives.

![Hierarchical Controller Diagram](image)

Drones must remain within a previously designated patrol zone and pursue intruders that try to cross it. The patrol zone is defined by a polygon, in which the vertices are GPS coordinates. Drones are configured with the mission-specific GPS coordinates prior to deployment. The drones are initially located on one of two base stations, to which they must eventually return in order to recharge their batteries. For the complete mission, we used a patrol zone with a size of 20,000 m by 500 m. Such a patrol zone would allow the coverage of the south coast of the Italian island of Lampedusa (see Figure 1), a major hub for illegal migration from Tunisia and Libya to Italy (Coppens, 2013).

For the complete task, we use three evolved behavior primitives: “Go To Waypoint”, “Patrol”, and “Pursue Intruder”. The behavior primitives were then combined in a mission controller using a top-level preprogrammed behavior arbitrator (see Figure 2). The evolution of the behavior primitives is discussed below.

**Evolved behaviors**

**Go to waypoint:** In many maritime missions, it is necessary for drones to move to a specific location, indicated by a GPS waypoint, without colliding with obstacles or nearby drones. The waypoints can be configured before the drone is deployed, or they can be discovered autonomously during a mission.

Controllers for the “Go To Waypoint” behavior were evolved using a single drone in an environment with an additional 50 static drones that posed as obstacles. The static drones were randomly placed between the starting position and the destination waypoint in a band with a width of 50 m, and the evolving drone had to reach a waypoint within 10,000 control cycles (equivalent to 1,000 seconds). The waypoint was placed from 500 m to 1,000 m from the drone’s starting position at a random orientation, with respect to the drone. The evaluation function rewarded controllers for (i) getting close to the waypoint, and (ii) stopping within 3 m of the waypoint. The evaluation function also penalized controllers for colliding with the static drones:

\[
    f = -10^{-2} \cdot P + \begin{cases} 
        \frac{D - d}{D} & \text{far from waypoint} \\
        2 - \frac{C - s}{C} & \text{near waypoint,} 
    \end{cases}
\]

where \( P \) is the number of control cycles that the drone collided with static drones, \( D \) is the initial distance from the drone to the waypoint, \( d \) is the current distance from the drone to the waypoint, \( C \) is the total number of control cycles, and \( s \) is the number of control cycles where the drone is stopped within 3 m of the waypoint. Seven of the ten evolutionary runs conducted for the “Go To Waypoint” behavior evolved solutions that shut down the motors after reaching the waypoint. In all seven successful solutions, the drone moves at full speed toward the waypoint while evading static drones. After arriving at the waypoint, the drone then shuts down the motors using the stop motors output of the ANN.
Figure 3: Evolved “Patrol” behavior with 50 drones in a patrol zone with a size of 300 m × 800 m at different times. In the figure, the size of the drones has been increased relative to the size of the patrol zone for clarity.

**Patrol:** Once inside the patrol zone, the drones should disperse to cover the zone evenly while remaining inside its pre-defined boundaries. Controllers were evolved for the “Patrol” behavior by placing 50 drones near the center of a rectangular zone with sizes varying from 250 m to 1,500 m from sample to sample. Controllers were evolved for 5,000 control cycles (equivalent to 500 seconds), and each drone had an individual copy of the evaluated ANN controller. The controllers were evaluated at each control cycle based on their distance to other drones and patrol zone boundaries, and on the energy spent:

\[
g_i = \sum_{r=1}^{R} \frac{\frac{2}{3} \cdot (1 - \max(\alpha_{ir})) + \frac{1}{3} \cdot (1 - p_{ir})}{R}
\]

where \(\alpha_{ir}\) is the set of activations of drone \(r\)’s boundary sensors and drone sensors at control cycle \(i\), \(p_{ir}\) is the percentage of power applied to the motors, and \(R\) is the number of drones. If any of the drones is outside the patrol zone at control cycle \(i\), the value of \(g_i\) is set to 0. The fitness obtained in a sample was the mean of \(g_i\) during the sample. The evolved controllers move outward in a circle until they stop sensing nearby drones. When the patrol zone is small and drones are able to reach its boundaries before the sample terminates, they stop moving outward and instead optimize the spacing between one another. An example of this behavior can be seen in Figure 3.

**Pursue intruder:** When an intruder is detected by a drone’s camera, the drone should attempt to follow it without crossing the boundaries of the patrol zone. Since the drones’ cameras are unable to collect depth data, a minimum of two drones should pursue an intruder in order to triangulate its position. For the evolution of the “Pursue Intruder” behavior, an intruder attempted to cross a patrol zone. The size of the patrol zone varied in each sample from 250 m to 1,000 m. The 50 drones, each with a copy of the controller being evaluated, started evenly dispersed inside the zone. The intruder moved at a fixed speed of 10 km/h. The controllers were evaluated according to the number of control cycles in which the intruder was detected. Furthermore, \(g_i\) was reused in order to reward dispersion of the drones, optimize power usage, and to obtain solutions in which the drones remain inside the patrol zone:

\[
h_i = g_i + \min(D_i, 2),
\]

where \(D_i\) is the number of drones detecting an intruder in control cycle \(i\). The fitness obtained in a sample was the mean of \(h_i\) during the sample. The evolved controllers pursue the intruder as soon as it is detected, and remain static otherwise. If several drones are pursuing the intruder, some of them cease the pursuit, typically leaving only two. As soon as the intruder exits the patrol zone, the drones stop the pursuit and adjust their positions with respect to neighboring drones.

**Preprogrammed arbitrator**

After the behavior primitives had been evolved, we combined them using a simple state-based preprogrammed arbitrator (see Figure 4). The arbitrator determines which behavior should be active at any given time, depending on the current state and the drone’s sensory inputs. When an intruder is detected, the drone alerts other drones within a 200 m radius. The alerted drones automatically activate the “Pursue Intruder” behavior. If a drone is in the “Pursue Intruder” behavior and has not seen an intruder for the past five minutes, it reverts to the “Patrol” behavior.

The amount of energy necessary to return to the base station is estimated based on the drone’s distance to the base...
Two or more drones pursuing an intruder has been detected after entering the patrol zone as we deploy more drones. The mean amount of time until a portion of time that two or more drones pursue an intruder decreases as more drones are deployed. With 100 drones, we can observe a slight improvement in the proportion of drones being pursued by either (i) one drone, or (ii) two or more drones. Ideally, an intruder should be pursued by at least two drones in order for its position to be correctly triangulated. Although the detection rate peaked at a swarm size of 200 drones, we have a battery life of ten hours of stand-by use, or two hours of full throttle. When back at the base station, the drones recharge their batteries for a period of 30 minutes. Two hours into the experiments, intruders begin to cross the patrol zone at random locations every 30 minutes (a total of 44 crossings). The simulation is run for a total of 24 simulated hours.

We ran experiments in ten different scenarios with a number of drones varying from 100 to 1,000 at increments of 100 drones. The simulation was run in a patrol zone with a size of 20,000 m by 500 m, as shown in Figure 1.

The performance observed for the different swarm sizes can be seen in Figure 5. By increasing the number of drones, the number of detected intruders increases until it reaches a maximum of 100% detection rate with 500 drones, which corresponds to a density of 50 drones/km². Another indicator of performance is the amount of time that the intruders are being pursued by either (i) one drone, or (ii) two or more drones. Ideally, an intruder should be pursued by at least two drones in order for its position to be correctly triangulated. Although the detection rate peaked at a swarm size of 500 drones, we can observe a slight improvement in the proportion of time that two or more drones pursue an intruder as we deploy more drones. The mean amount of time until an intruder has been detected after entering the patrol zone decreases as more drones are deployed. With 100 drones the mean amount of time was 62 s, with 500 drones the mean amount of time was 19 s, and with 1,000 drones the average time was 16 s.

In the scenario with 500 drones, the hybrid controllers successfully completed the task by detecting 44 out of the 44 intruders. In total, intruders were present in the patrol zone for 132 minutes, of which they were detected by a single drone for 28 minutes, and by two or more drones for 92 minutes (21% and 70% of the total time, respectively). In the final ten hours, the system reached an equilibrium in which a mean of 79% of the drones were patrolling, 14% were returning to the base or recharging, 6% were going from the base to the patrol zone, and 1% were pursuing an intruder (see Figure 6). We conducted the same analysis for the setup with 1,000 drones and observed similar results.

**Conclusions and Future Work**

In this paper, we applied a hybrid approach, based on a combination of evolved and preprogrammed control, to the synthesis of controllers for large swarms of aquatic drones. Controllers for three different behavior primitives were evolved: “Go To Waypoint”, “Patrol”, and “Pursue Intruder”. The controllers were then combined using a state-based preprogrammed behavior arbitrator in order to execute a patrolling and intruder detection task.

The use of ER techniques for the synthesis of self-organized control is particularly relevant for large-scale multirobot systems based on decentralized control, since it is often challenging to manually design behavioral rules for the individual robots. Evolutionary techniques, however, are difficult to apply in complex tasks, provide little control over the solutions evolved, and require a considerable amount of time to evolve controllers for large robotic swarms. By dividing the task into simpler sub-tasks, it is possible to evolve robust and scalable behaviors that solve different parts of the
task. The controllers evolved for the sub-tasks can then be combined using a preprogrammed behavior arbitrator that controls when each behavior primitive should be executed. Furthermore, evolved behavior primitives can potentially be reused in various types of missions.

In our ongoing work, we are exploring ways in which inter-drone communication can be further exploited in maritime scenarios. Moreover, we are preparing our hardware testbed for proof-of-concept missions based on the approach proposed in this paper. Our long-term goal is to have a software and hardware platform for swarm-based maritime missions.

Acknowledgements

This work was partly supported by FCT – Foundation of Science and Technology under grants SFRH/BD/76438/2011, PEst-OE/EEI/LA0008/2013 and EXPL/EEI-AUT/0329/2013.

References


