# Profiling Underwater Swarm Robotic Shoaling Performance using Simulation

Mark Read[1,*], Christoph Möslinger[2], Tobias Dipper[3], Daniela Kengyel[2], James Hilder[1], Ronald Thenius[2], Andy Tyrrell[1], Jon Timmis[1], and Thomas Schmickl[2]

[1] Department of Electronics, the University of York, UK.
[2] Artificial Life Laboratory, the University of Graz, Austria.
[3] Institut für Parallele und Verteilte Systeme, Universität Stuttgart, Germany.
∗ *mark.read@york.ac.uk*

**Abstract.** Underwater exploration is important for mapping out the oceans, environmental monitoring, and search and rescue, yet water represents one of the most challenging of operational environments. The CoCoRo project proposes to address these challenges using cognitive swarm intelligent systems. We present here CoCoRoSim, an underwater swarm robotics simulation used in designing underwater swarm robotic systems. Collective coordination of robots represents principle challenge here, and use simulation in evaluating shoaling algorithm performance given the communication, localization and orientation challenges of underwater environments. We find communication to be essential for well-coordinated shoals, and provided communication is possible, inexact localization does not significantly impact performance. As a proof of concept simulation is employed in evaluating shoaling performance in turbulent waters.

## 1 Introduction

The ocean remains the least explored habitat on earth, hosting undiscovered organisms and resources of interest and value. The importance of addressing underwater search and environmental monitoring is exemplified by events such as the 2010 BP Deepwater Horizon oil spill and the 2009 Air France Flight 447 crash in the Atlantic ocean, where it took nearly 2 years to recover the black boxes from the ocean floor. Water is an extremely challenging environment to operate within, visibility is poor, and electromagnetic signals are heavily attenuated, complicating communication and GPS-based localization. The CoCoRo project[1] seeks to advance underwater exploration capability through use of swarm intelligent systems endowed with collective cognitive decision making abilities [11]. Collective cognition is intended to assist swarms in coping with a noisy and heterogeneous environment, identifying and discriminating between multiple underwater targets, dynamically reallocating robots between tasks to meet requirements, compensating for failed or lost swarm members, and maintaining a communication network of robots between an exploratory swarm and the water surface.

[1] The EC funded CoCoRo Project, GA 270382; http://cocoro.uni-graz.at/

Engineering swarm robotic systems is an inherently challenging field: the robotic platforms are complex, as are the environments in which they are deployed, and group behaviours must be engineered through the manipulation of interactions between individuals. As such computational simulation is frequently employed to aid in research, development and evaluation; the joint SYMBRION-REPLICATOR projects have developed a sophisticated 3D robotics simulation, Symbricator3D [13]. Symbricator3D employs highly realistic sensor and actuator models, and robotic controllers developed on the simulation should migrate directly onto real platforms. However it is a highly complex piece of software, and its documentation is lacking in comparison to other simulations reviewed here. The Jasmine swarm robotic platform is accompanied by a simulation of the platforms[2]; this software simulates robots on a 2D plane. Simbad is a recently developed 3D robotics simulation written in Java with a slant towards evolutionary robotics research [6]. Documentation is of a high quality, with a javadoc API published online along with details of the simulation's architecture and some tutorials. The Stage simulation, developed under the Player/Stage project, has been widely adopted for swarm robotic research and educational purposes [4, 5]. Stage is however restricted to simulating 2D environments, and in addressing 3D environments the Gazebo simulation was developed [8]. Gazebo encompasses a realistic physics engine, where simulated bodies have properties such as mass, friction and bounce factors. However, due to its computational load Gazebo can simulate at most 10 robots, limiting its application in swarm robotic systems.

In this paper we present the CoCoRoSim underwater robotics simulation, developed by the CoCoRo project to facilitate controller conceptualization, development and evaluation. CoCoRoSim is implemented in NetLogo 3D, and as such can simulate large numbers of robots. Its Newtonian mechanics physics engine simulates water drag forces, buoyancy, translational and rotational motion, a variety of robotic sensor systems, and forces exerted on simulated bodies by water currents. We present CoCoRoSim in section 2, and its calibration against a CoCoRo robotic platform in section 4. CoCoRoSim's use in adapting a generic shoaling algorithm to the constraints of underwater environments and evaluating its performance is detailed in section 5. T his section also explores how CoCoRoSim can be used to evaluate controller performance in the presence of water currents and turbulence. Lastly, section 6 concludes this paper.

## 2 The CoCoRoSim simulation

The CoCoRoSim simulation is implemented in NetLogo 3D[3], a three dimensional multi-agent modelling environment. The role of CoCoRoSim in the CoCoRo project is as a fast means of prototyping algorithms and AUV deployment scenarios; the CoCoRo project's ethos is to migrate ideas to real-world robots as quickly as possible following their validation in simulation. As such, NetLogo 3D was selected as an implementation platform: it provides a powerful integrated

[2] *http://www.swarmrobot.org/Simulation.html*
[3] *http://ccl.northwestern.edu/netlogo/*

(a) The Lily platform.

(b) The Jeff platform, from the side (top) and top (below).

Fig. 1: Two underwater robotic platforms developed in the CoCoRo project.

3D GUI interface, is well supported and documented, and using behaviorspace can be executed on a computation cluster to facilitate large-scale experimentation. The language is well suited for non-programmers on the project to pick up quickly with minimal effort, and its features have aided in minimising development time, hence allowing a greater focus on algorithmic development and real-world deployment. CoCoRoSim can be downloaded from *http://cocoro.uni-graz.at/drupal/media* under 'Software'.

### 2.1 Sensors, actuators, operating system and robots

CoCoRoSim provides representations of the sensors and actuators developed and used within CoCoRo, and allows these to be configured and calibrated to reflect the autonomous underwater vehicles (AUVs) developed in the project (figure 1).

Lily is a modified toy submarine, used to facilitate fast hardware and controller prototyping. It has a relatively short operational life-span. In contrast, Jeff is developed and manufactured over a longer term; it is more robust, maneuverable and has a greater number of more powerful sensor and actuator systems. Both Lily and Jeff platforms are equipped with blue light sensor systems (BLSS), each of which comprises multiple LEDs and a receiver. Lily has 5 BLSS systems whereas Jeff has 6. BLSS receivers observe blue light within a cone of 120°. They can provide communication through the pulsing of LEDs, can be used for distance sensing, and are capable of detecting obstacles by observing LED reflections. BLSS are implemented in CoCoRoSim by registering the nearest AUVs within a cone of observation, centred according to the BLSS configurations on the AUVs. Simulated BLSS detect proximity to obstacles such as tank walls and floor by tracing rays projected along the centre and limits of the cone of observation. This implementation represents a compromise between computational efficiency and accuracy. BLSS communication is implemented through direct message passing to the nearest AUV within a sensor's cone.

CoCoRo AUVs are equipped with radio transmitters and receivers to provide omni-directional communication. These are implemented by direct message

passing between AUVs of sufficient proximities. Pressure sensor, compass, accelerometer and gyroscope implementations are also provided.

Lily and Jeff platforms manoeuvre with propellor-based thrusters. These are simulated as simply providing a force of movement on the AUV along the axis in which they are oriented. Vertical manoeuvrability is provided by a buoyancy pump which changes the volume of the AUV. In CoCoRoSim such actuators are implemented as providing a vertical (in relation to the environment) force on the AUV based on its density in relation to water.

Controller interaction with CoCoRo AUV platforms is provided through an operating system based on FreeRTOS [1]. This OS's functionality is reflected in CoCoRoSim, with semantically similar functions being provided to ease migration of simulation algorithms to real-world platforms. The OS provides functions that allow, for example, controllers to specify a particular depth or heading and speed, and through the use of simple PD controllers the platform's actuators are manipulated to this end. The PD controller for heading adjusts the differential in thruster settings around a specified percent of backwards or forwards maximum thrust to maintain some specified forward speed and a particular heading based on compass readings.

## 3 Physics engine

The CoCoRoSim physics engine discretizes time; the states of all simulated bodies are updated in each time step. The physics engine implements Newtonian mechanics with simulated bodies modelled as particles. The particles' translational and rotational velocities and accelerations in the $x$, $y$ and $z$ axes are influenced by forces acting on the body. There is currently no provision for simulated bodies to change their pitch or roll.

The translational drag force of water $F_{dt}$ on a simulated body moving at velocity $v$ is modelled using equation 1.

$$F_{dt} = 0.5 \cdot \rho \cdot v^2 \cdot A \cdot C_d \tag{1}$$

Where $\rho$ represents the density of water, 1000 kg/m$^3$. $A$ represents the cross sectional area of body in the direction of its movement through the water, and $C_d$ represents the drag coefficient of the body. $A \cdot C_d$ is specific to the body, and these values are calibrated using the procedure outlined in section 4.

Rotational acceleration and rotational velocity are modelled as:

$$a_r = \tau / I \quad \text{and} \quad \omega = a_r \cdot \delta \tag{2}$$

Where $a_r$ is rotational acceleration in radians/s$^2$, $\tau$ is the net torque acting on the AUV, and $I$ is the mass moment of inertia. $\tau$ is the sum of torque delivered by the thrusters, and rotational drag forces. $\omega$ represents rotational velocity in radians/second, and $\delta$ is the length of time represented by a simulation timestep. Rotational movement is countered by a drag torque $\tau_d$ arising from form drag and skin frictions which have been abstracted into a single parameter $C_r$ that can be calibrated for a particular AUV:

$$\tau_d = \omega^2 \cdot C_r \tag{3}$$

Vertical movement of the AUV is dictated by the net forces of the buoyancy pump and gravity, and are subject to translational drag as in equation 1. A submerged AUV that is stationary in the vertical plane exhibits an equilibrium between gravity acting on the AUV's mass and the upwards force resulting from its buoyancy. Hence, the net vertical force, $F_v$, is:

$$F_v = (m \cdot g) - (\rho \cdot V \cdot g) \tag{4}$$

where $\rho$ is the density of water, 1000, and $m$ and $V$ represent the AUV's mass and volume respectively. $g$ represents acceleration under gravity. CoCoRoSim models AUV buoyancy pumps as changing an AUV's volume.

### 3.1 Simulation of water currents

CoCoRoSim can simulate water currents, represented as forces along each axis at points in discretised space (patches), that influence AUV motion. They are generated by AUV thrusters, and by 'cold-spots': low pressure patches that generate current forces towards them. Currents are subject to diffusion and to decay.

AUV thrusters suck water into them and propel it out the back, as depicted in figure 2a. The current force being pulled into the thrusters is equal to that pushed out, and is equivalent to the force the thrusters apply to the AUV. Each thruster's forces are considered individually. The left thruster creates a current towards the AUV in the patch to the left of it. An equal force is created behind the AUV: one fourth of this is created in each of the patches directly behind, behind and up, behind and down, and behind and left the AUV. The same applies, vice versa, to the right thruster.

A cold spot creates forces directed towards it in the patches surrounding it. They persist for a period of time, with the current forces they generate rising and falling in magnitude over this time to prevent any excessive force differentials being generated from seemingly nowhere. The periodic turnover of cold spots in randomly selected patches throughout the simulation creates dynamic convections in the water.

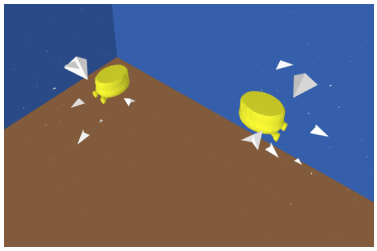
Current forces are subject to logarithmic decay and to diffusion; each time step the change in current component of each patch, $F_c$, is:

$$\Delta F_c = \Big( \big( (1 - \kappa) \cdot \text{avg}_\text{N}(F_c) \big) - (\kappa \cdot F_c) \Big) \cdot \gamma \tag{5}$$

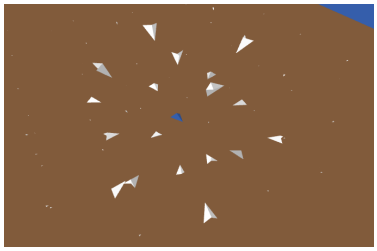
where $\kappa$ holds a value between 0 and 1 and represents the rate at which forces diffuse; $\text{avg}_\text{N}$ is a function returning the mean of that current component amongst the neighbouring 26 patches[4], and $\gamma$ is the decay rate.

Current forces affect both translational and rotational AUV movements (figure 2c). When calculating the net translational forces that dictate AUV acceleration, the forces on the patch that the AUV occupies are considered. The differential between opposing patches in the 4-patch neighbourhood are used to calculate a torque when calculating AUV rotational acceleration.

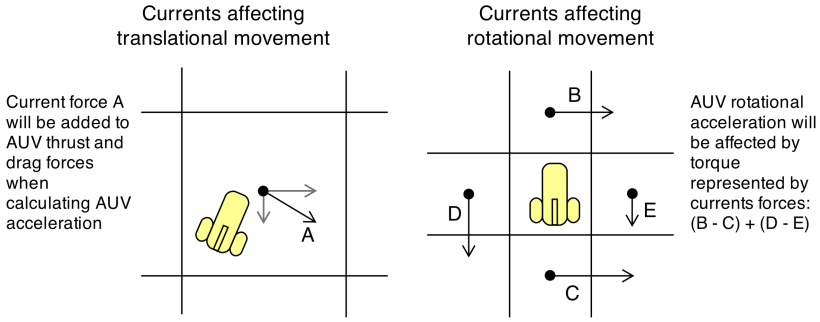
[4] Or less if the patch being updated lies on the environment boundary.



(a) Currents generated by AUV thrusters.



(b) Cold-spots (blue arrow) act as sinks for current forces.




(c) Current forces in the same patch as an AUV, and on the 4-neighbourhood patches around it affect its translational and rotational accelerations.

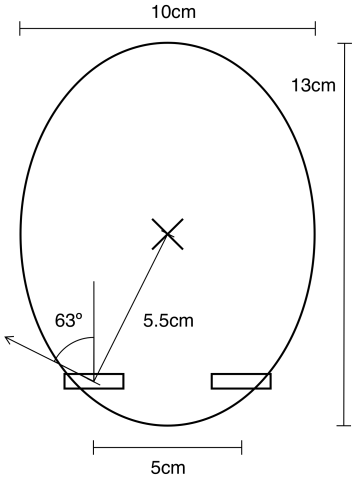
Fig. 2: Currents are represented as forces in three dimensions in discretised space.

## 4 Calibration

This section reports the calibration of CoCoRoSim's representation of the Lily platform (figure 1a). The Jeff platform is still under development, and has not yet been calibrated in CoCoRoSim; when it is complete, it will undergo a similar calibration process in CoCoRoSim.

The Lily AUV weighs 0.44kg. It has two thrusters capable of propelling the AUV forward, and has a buoyancy pump that provides vertical motion. Lily is unable to change its pitch or roll. The two thrusters have been empirically measured as able to deliver a combined maximum force of 0.01N, through their ability to lift a weight (shown in figure 3a). The magnitude of the friction force between the string and the pipes over which it runs was not known, and as such two variations of the experiment were performed. Firstly the AUV is held by hand, with the string taught, and then released. Secondly, the string is slack, and the AUV accelerates to pull it taught. The direction of the frictional force on the string is opposite in the two experiments, which gave readings of the AUV lifting 0.8g and 1.2g respectively. The AUV was assumed to have lifted 1g when

(a) Empirically measuring Lily maximum thruster force in the tank, based on its ability to lift a weight.




(b) The orientation of thrusters on Lily, and their location with respect to the centre of rotation (marked by an X).

Fig. 3: The calibration of Lily in CoCoRoSim.

calculating AUV maximum thrust force as being 0.01N. Hence, each individual thruster can deliver 0.005N of thrust.

Lily's translational cross-section and drag coefficient, $A \cdot C_d$ in equation 1, are difficult to calculate. Instead, they have been deduced based on maximum thrust and terminal velocity, given that:

$$v_t = \sqrt{\frac{2F_{dt}}{\rho \cdot A \cdot C_d}} \quad \text{rearranges to:} \quad A \cdot C_d = \frac{2F_{dt}}{\rho \cdot v_t^2} \tag{6}$$

Lily's terminal velocity has been empirically measured at 7.5cm/s, and given its maximum thruster force ($F_{dt}$) of 0.01N, $A \cdot C_d$ is 1.78. This provides simulated behaviours corresponding with empirical measurements of Lily's acceleration: from a standing start at full thrust Lily moves 10cm in 3.03 seconds.

Using the buoyancy pump Lily can change its volume between 430 and 450cm$^3$. A value of 440cm$^3$ delivers a net vertical force $F_v = 0$. When Lily has a volume of 450cm$^3$, the net force $F_v = 0.1$N. Hence, the buoyancy pump can deliver a vertical force of $\pm 0.1$N. The change in density of water over Lily's 3m diving limit is deemed negligible. As such, the buoyancy pump is not used to set a desired depth directly, but the buoyancy force, and hence the speed at which an AUV descends or ascends.

Rotational torque is delivered through the differential between the AUV's thrusters. Empirical measurements taken of Lily in the water reveal that the centre of rotation is the centre of the AUV, despite the thrusters being offset from this point, as shown in figure 3b. The rotational torque resulting from the

thrusters, $\tau_T$, is calculated as:

$$\tau_T = (T_l - T_r) \cdot 0.055 \cdot \cos(63°) \tag{7}$$

where $T_l$ and $T_r$ represents the thruster force exerted by the left and right thrusters respectively, the thrusters are 0.055m from the centre of rotation, and oriented 63° from the perpendicular through which torque is applied. In calculating rotational drag, $C_r$ can be estimated given Lily's terminal rotational velocity, empirically measured as 1.48 rad/s. At rotational terminal velocity, $\omega_t$, the torque provided by the thrusters is equal to the opposing torque originating from rotational drag forces. Using an equation of the form of translational terminal velocity (equation 6 above), the following equation describing terminal rotational velocity is formed:

$$\omega_t = \sqrt{\tau_{max}/C_r} \tag{8}$$

Where $\tau_{max}$ represents the maximum torque the thrusters can deliver. Rearranging to solve for $C_r$:

$$C_r = \big((0.005 - -0.005) \cdot 0.055 \cdot \cos(63°)\big)/\omega_t^2 \tag{9}$$

Which solves to give $C_r = 0.000114$. Lily's mass moment of inertia, $I$, is not known. However, given that the coefficient of rotational drag and maximum thruster forces are known, it can be calibrated in simulation to deliver similar rotational accelerations to those empirically observed of Lily. A value of $I = 0.0005$ is used, as this matched observations that from a standing start Lily can rotate 90° in 2.4 seconds, 180° after 3.7 seconds, 270° after 4.8 seconds, and completes a full turn after 5.9 seconds.

## 5 Profiling controller behaviour

This section demonstrates CoCoRoSim's use in controller design and evaluation. A principle challenge in underwater swarm robotics is collective motion; AUVs must be coordinated to efficiently explore the environment and not get lost in the ocean. Reynold's Boids algorithm is popular in computational simulations requiring collective motion [10], for example coordinating dinosaurs and bats in Jurassic Park and Batman Returns movies. Swarm member (termed a 'boid') motion is dictated by three rules: *cohesion* attracts boids to their neighbours, *separation* prevents them from colliding, and *alignment* promotes common velocities within the group [10]. We report here preliminary investigations into the suitability of the Boids algorithm for deployment on CoCoRo AUVs.

The principle challenges in real-world deployment are localization and communication. Lily AUVs can localize one another through use of blue light systems, which have a range of around 50cm, and can detect distances to other AUVs only within a cone of observation (120°); exact triangulation is not possible. Furthermore it is likely that only the nearest neighbour will be detected. Communication is provided through omni-directional radio-frequency (range 50cm), or directional-blue light systems where the nearest neighbour is the only likely recipient of a message. We have performed a series of experiments to

examine how Boids's performance in underwater shoaling is effected by these constraints (figure 4). The *Vanilla* (Van) experiment refers to boids employing exact triangulation, each boid knows the exact location of all neighbours within 50cm. This is impossible in real AUVs, but is performed in simulation as a baseline against which to examine performance of various adaptations of Boids on CoCoRo AUVs. In the *blue light triangulation* (BLT) group boids can only detect whether or not an AUV lies within a 120° cone, and the distance within that cone. If detected a neighbour is assumed to lie in the centre of the cone. Boid velocities are communicated omni-directionally to all neighbours within 50cm using radio frequency. *Blue light communication* (BLC) extends BLT by communicating velocities only with the nearest neighbours over blue light, this leaves radio frequency communication free for other tasks CoCoRo shoals will have to perform. Lastly, because communication underwater is problematic and it will likely be needed for other swarm functions, *no alignment* (NA) examines shoal performance in absence of any velocity communication, effectively nullifying the alignment rule. Both BLC and BLT simulated algorithms assume noise-less loss-less instantaneous communication, implemented through direct message passing. Given Lily's relatively short sensor range in contrast to its terminal velocity, these algorithms were limited to using only 10% of maximum thruster force to prevent erratic shoaling behaviour.

Six metrics of shoaling performance are applied, and experiments are conducted with 11 AUVs in total. Polarisation measures the degree to which all boids are pointed in the same direction, calculated as in [7]. A polarisation of 1 indicates that all shoal members have the same orientations, whereas 0 indicates a uniform spread of orientations; higher values are desirable. Angular momentum measures the degree of shoal rotation around its centre, calculated as in [7]. This measure complements polarisation: a shoal of boids rotating clockwise around some point can have a very low polarisation, yet high angular velocities indicate a shoal that is still well organized. Shoal speed measures the movement of the shoal's centre, a highly motile shoal is desirable. The number of times that an AUV is lost from the shoal is counted, and the mean number of distinct shoals throughout simulation time is also recorded. Fewer lost AUVs and low numbers of shoals are desirable. Shoal separation represents the median separation between AUVs in the shoal over the entire simulation time. Algorithms that can provide both wide and narrow separations are desirable, provided no more AUVs are lost. These metrics are shown as box plots in figure 4. The magnitude of effect change in comparison with the Vanilla experiment is calculated using the Vargha-Delaney A test [12], a non-parametric effect magnitude test that calculates the probability that a randomly selected sample from population A is larger than a randomly selected sample from population B. Values of $\geq 0.71$ or $\leq 0.29$ are assumed 'large'. Each experimental group comprises 400 simulations/samples, the number required to reduce the effect of stochastic variation on results to a "small" effect (procedure described in [9]). The boid's cohesion, separation and alignment weights were assigned values of 1.0, 1.0 and 10.0 respectively, a separation threshold of 15cm was used, and boids had a full 360°

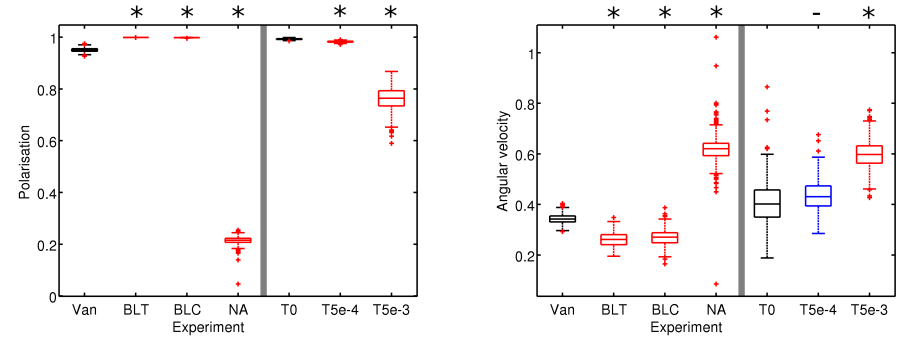
vision around them. These values were identified through preliminary experimentation to give stable shoaling behaviour. The simulation was executed for 25,000 time steps, which represents 12,500 seconds. The simulated environment was 6m x 6m with a 2m depth.

A switch from the vanilla group's perfect triangulation to blue-light based triangulation (BLT and BLC groups) does not deteriorate shoal polarisation, in fact it is increased. This is reflected in a reduced angular velocity in the BLT and BLC groups. This result is unexpected, as blue light-based triangulation is less precise. It may be explained through blue light's consideration of only the nearest neighbours, and boids thus perceiving swarm centres to be closer than they in fact are. Hence the impetus to turn towards the centre is reduced in contrast to the influences of alignment and separation. This may also explain why BLT and BLC groups have significantly reduced separations in contrast to the vanilla group: as demonstrated by the no alignment (NA) group, a lack of alignment results in a more spread out shoal. The BLT group has a significant reduction in the number of AUVs that lose the shoal, and in the number of shoals that emerge. However this change is lost in the BLC group where velocities are shared with only the nearest neighbours. It is clear from these results that no significant detriment to shoal quality occurs when adapting the vanilla boids algorithm to the constraints of CoCoRo platforms. It is also clear that although communication on these platforms is a scarce resource, reserving it purely for higher shoal functions at the expense of communicating velocities has a significantly detrimental impact on performance: the NA shoals were less polarised, had lower velocities, and lost more AUVs.

### 5.1 Controller performance in turbulent waters

CoCoRoSim's simulation of current forces in the water permits analysis of controller performance in turbulent waters. This is demonstrated by examining the deterioration of the BLT boids variant's shoaling performance in increasingly turbulent waters. These experiments are run for 5,000 simulated time steps, representing 2,500 seconds, in a tank of $5 \times 5 \times 2$m. A smaller simulated environment and shorter runtime were selected to address the considerable computational requirements of simulating currents. Three experiments are conducted, with turbulence (represented by the maximum force a cold spot can exert) set at 0N (*T0* in figure 4), 0.0005N (*T5e-4*) and 0.005N (*T5e-3*). The probability of any patch of discretised space becoming a cold spot in a time step is 0.0015, and cold spots persist for 12.5 seconds. 60% of the current force in a patch is diffused to surrounding patches every time step, and the decay rate is set to 0.65. These values have not been calibrated to any particular body of water, they were selected on the basis that a 0.5kg floating simulated body exhibited visually appropriate trajectories. These experiments are to demonstrate that CoCoRoSim can be used to evaluate controller performance in the presence of water currents.

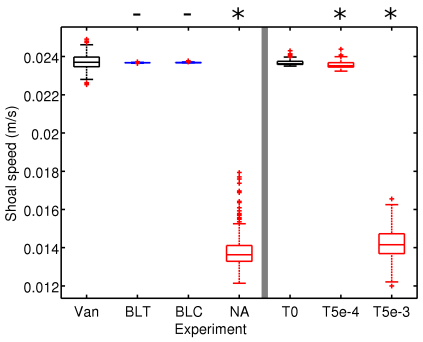
As shown in figure 4, shoaling performance is significantly altered in only 2 of the 6 metrics for current sink forces of 0.0005N (*T5e-4*), and these changes are comparatively minor in contrast with forces of 0.005N (*T5e-3*) where all metrics



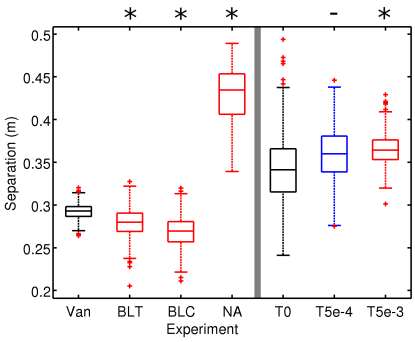

(a) Shoal polarisation



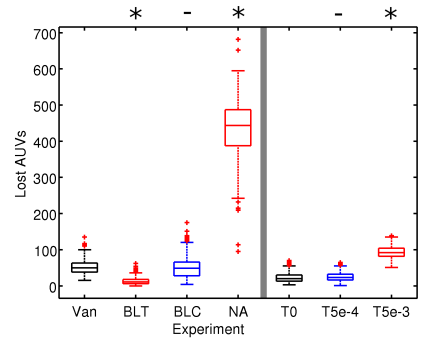

(b) Shoal angular velocity



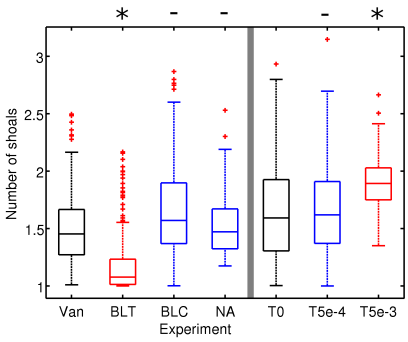

(c) Shoal speed




(d) Shoal separation




(e) Lost AUVs




(f) Number of shoals

Fig. 4: The shoaling performance of various adaptations of boids and in the presence of various levels of water turbulence. Experiments left of grey vertical lines represent variations of boids, those to the right represent the BLT algorithm running in the presence of water currents and varying degrees of turbulent water. Effect magnitude is established through the A test [12], where red ($*$) and blue ($-$) boxes indicate the presence or absence of large effect magnitudes with respect to the vanilla or T0 algorithm (depending on the experiment).

reveal significant changes. Although Boids could use at most 10% of available thrust, the OS may make full use of the thrusters in attempting to maintain a particular heading. It is perhaps unsurprising that turbulence forces of 0.01N, Lily's maximum thrust force, cannot be tolerated. However, it is notable that the shoaling algorithm tolerates, without significant deterioration of performance in many cases, turbulences of 10% of the maximum available thrust.

## 6 Discussion and further work

This paper has presented the CoCoRoSim underwater swarm robotics simulation, detailed its calibration against empirical data taken from a real-world robot, and demonstrated its use in profiling robot controller behaviour. Collective coordination in underwater swarms is a challenging task owing to low visibility, the attenuation of electro-magnetic signals typically used for communication, and current forces inherent in moving water. We have made use of simulation in evaluating the quality of shoaling resulting from adapting Reynold's Boids algorithm to these underwater constraints. CoCoRo AUVs have limited communication bandwidth, and this might ideally be reserved for higher swarm functions when addressing a complex task of which shoaling is only one constituent activity. However, the presented results clearly indicate that without communication the coordination of shoals using Boids is problematic. Surprisingly, given the ability to communicate velocities amongst shoal members, the limited observational and triangulation capacity of CoCoRo AUVs does not have as detrimental affect on coordination as might be expected. There exist other flocking algorithms that are inherently communication-less, for example [3, 2], however this algorithm has not been investigated in platforms with the present number of degrees of freedom. Investigating the suitability of such algorithms in controlling underwater shoals is left as further work, as is the deployment of the present shoaling algorithm on real-world hardware.

Simulation can substantially aid underwater systems research and development. The impracticalities of developing and calibrating algorithms on real-world robotic platforms that must be removed from the water, opened for reprogramming, charged, sealed, and redeployed prior to evaluation are prohibitive. CoCoRoSim facilitates must faster initial development of algorithms. Detailed profiling and assessment of shoaling performance, as performed here in simulation, would not be possible in the real platforms. Furthermore, CoCoRoSim permits the environmental limits within which controllers can reliably operate to be ascertained; we have examined how shoaling performance degrades in increasingly turbulent waters. This is provided as a proof of concept, calibration of this water currents simulation framework against a particular body of water is left as further work. Simulation of water currents will lead into simulating the movement of chemical plumes, useful in designing swarm-intelligence algorithms capable of performing underwater plume tracking.

## References


1. R Barry. The FreeRTOS Reference Manual. *freertos.org*, 2011.
2. C Moeslinger *et al.* A minimalist flocking algorithm for swarm robots. In *Proceedings of ECAL '09, Springer LNCS series 5778*, pages 375–382, 2009.
3. C Moeslinger *et al.* Emergent flocking with low-end swarm robotics. In *Proceedings of ANTS '10, Springer LNCS 6234*, pages 424–431, 2010.
4. B Gerkey, R Vaughan, and A Howard. The player/stage project: Tools for multi-robot and distributed sensor systems. In *Proceedings of the International Conference on Advanced Robotics*, pages 317–323, 2003.
5. H Lau *et al.* Adaptive data-driven error detection in swarm robotics with statistical classifiers. *Robotics and Autonomous Systems*, 59(12):1021–1035.
6. L Hugues, N Bredeche, and TI Futurs. Simbad: an autonomous robot simulation package for education and research. In *in Proceedings of SAB '06. Springer LNAI series 4095*, pages 831–842, 2006.
7. I Couzin *et al.* Collective memory and spatial sorting in animal groups. *Journal of Theoretical Biology*, 218:1–11, 2002.
8. N Koenig and A Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *In IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2149–2154, 2004.
9. M Read *et al.* Techniques for grounding agent-based simulations in the real domain: A case study in experimental autoimmune encephalomyelitis. *MCMDS*, 17(4):296–302, 2011.
10. C Reynolds. Flocks, herts and schools: A distributed behavioral model. *Computer Graphics*, 21(4):25–34, 1987.
11. T Schmickl *et al.* Cocoro - the self-aware underwater swarm. In *Proceedings of the SASO '11*, 2011.
12. A Vargha and HD Delaney. A critique and improvement of the cl common language effect size statistics of McGraw and Wong. *Journal of Educational and Behavioral Statistics*, 25(2):101–132, 2000.
13. L Winkler and H Wörn. Symbricator3D - A Distributed Simulation Environment for Modular Robots. In *Proceedings of ICIRA '09*, pages 1266–1277, 2009.