Cooperative Autonomous Marine Vehicles for Adaptive Passive Acoustic Monitoring

Rossides, George

Award date:
2022

Awarding institution:
University of Bath

Cooperative Autonomous Marine Vehicles for Adaptive Passive Acoustic Monitoring

submitted by

George Rossides

for the degree of Doctor of Philosophy

of the

University of Bath

Department of Electronic and Electrical Engineering

December 2021

Swarm behavior ... emerges naturally from simple rules of interaction, as happens with a wave generated by a crowd of spectators at a football game.

The wave might look to a visiting Martian like a complicated exercise in logistics, but its dynamic pattern emerges from a simple rule: stand up and stick your hands in the air as soon as you see your neighbor doing it.

LEN FISHER – The Perfect Swarm (2009)

Acknowledgements

Firstly, I would like to thank my doctoral supervisors, Dr. Benjamin Metcalfe and Dr. Alan Hunter for their continuous guidance and support throughout this whole experience. This work would have not been possible without their knowledge, patience and great amount of time spent on guiding me and it has been a great pleasure working with them.

Additionally, I would like to thank the organisers, management team and people of the NEXUSS CDT program of the University of Southampton for funding this work and providing me with valuable training, networking opportunities and overall good experiences.

Finally, I would like to thank my family and friends for their continuous support. Among them special thanks should go to my friend and fellow student Stefan Chindea for his invaluable help at all difficult times. Moreover I would like to thank my housemate Marios Mouzouras and my friends Panos Koulountzios and Rafaella Antoniou for always being there until the very end.

Summary

In the quest for the exploration and study of the vast and complex oceanic environments and the biological species that inhabit them, a variety of different methodologies and techniques are currently employed. Among them, the ability to localise and monitor underwater acoustic sources by passively observing the signals that they emit was proven to be a valuable asset for the study of both biological entities (e.g. marine mammals) as well as anthropogenic ones (e.g. transportation vessels, off-shore drilling processes etc.). Traditional marine environment monitoring systems employ a single oceanographic research vessel, equipped with all the necessary sensors for the acquisition of data. This method of study of marine environments is expensive, time-consuming and unable to study efficiently the large areas that need to be explored. Therefore, recent research turned towards the employment of low-cost, semi-disposable, distributed sensing nodes for simultaneous data collection over large distances. The large numbers of nodes used in such systems, combined with their simple, low-cost, decentralised nature matches the ethos of another engineering field - that of swarm robotics. Therefore, it has recently been proposed that swarm intelligence algorithms and techniques could be used for the control of the movement of such marine systems.

This thesis introduces several novel swarm intelligence algorithms for use for the control of marine distributed sensor networks with passive acoustic capabilities. This is achieved through the modification of a prevalent swarm intelligence algorithm, *particle swarm optimisation* (PSO), originally used in numerical optimisation applications. The first half of the thesis is concerned with adapting PSO for the accurate low-level motion control of robotic swarms with the task of source localisation, while also allowing for the inclusion of alternative swarm robotic tasks like obstacle avoidance and aggregation. Afterwards, the thesis focuses on combining PSO with wavefield correlation techniques, currently used in marine passive acoustic systems such as multi-hydrophone arrays. In the end, it is demonstrated how the proposed algorithms can be combined together, enabling a robotic swarm of marine surface vehicles to localise a marine acoustic source, approach and encircle it and continue monitoring it while maintaining a minimum distance from it. All of the presented algorithms are implemented and tested using MATLAB simulations, while some of them are also further validated using Gazebo simulations employing detailed robot models.

Contents

1 Introduction **18**

1.1 Thesis Motivation 18

1.2 Covid-19 Pandemic Impact 20

1.3 Contributions 20

2 Introduction to Swarm Robotics **24**

2.1 Swarm Intelligence Algorithms 27

2.1.1 Particle Swarm Optimisation 28

2.1.2 Ant Colony Optimisation 31

2.1.3 Artificial Bee Colony Optimisation 31

2.1.4 Glowwarm Swarm Optimisation 31

2.1.5 Firefly Algorithm 32

2.1.6 Potential Fields 33

2.1.7 SIA Applications to Swarm Robotics 35

2.2 Swarm Robotic Tasks 41

2.2.1 Aggregation 43

2.2.2 Flocking 43

2.2.3 Target Entrapment 46

2.2.4 Multi-Target Tracking 47

2.3 Discussion of Current Literature 48

2.4 Conclusion 50

3 Adapted Particle Swarm Optimisation **58**

3.1 Particle Swarm Optimisation Theory 59

3.1.1 Parameter Tuning 60

3.2 Adaptation of PSO for Swarm Robotics 62

3.2.1 Updated Parameter Stability Criteria 64

3.3 Control of Velocity and Acceleration . . . 65
3.3.1 State model . . . 66
3.3.2 State Space Analysis . . . 68
3.3.3 Derivation of Extreme Cases . . . 70
3.4 Guidelines . . . 73
3.5 Simulations . . . 74
3.5.1 Constant Parameters . . . 75
3.5.2 Variable Parameters . . . 77
3.6 Discussion . . . 79
3.7 Generalised Adapted PSO . . . 80
3.8 Conclusions . . . 81

4 Obstacle Avoidance and Dynamic Velocity Control Strategies 85
4.1 Robotic Particle Swarm Optimisation . . . 86
4.2 Adapted Robotic Particle Swarm Optimisation . . . 88
4.2.1 Calibration of Individual Accelerating Terms . . . 89
4.2.2 Implementation of Dynamic Velocity Control Strategy . . . 90
4.3 Simulations . . . 92
4.3.1 World description . . . 95
4.3.2 Robot description . . . 96
4.3.3 Gazebo Simulations . . . 98
4.4 Results . . . 102
4.5 Discussion and Comparison with other SIA . . . 104
4.6 Conclusions . . . 107

5 Robot-Centred Reference Frame and the Non-Omnidirectional PSO Controller 109
5.1 Robot-Centred Reference Frame . . . 110
5.1.1 Robot-Centred Reference Frame Generalised Adapted PSO . . . 111
5.2 Obstacle Avoidance and Dynamic Velocity Control . . . 114
5.2.1 Non-Omnidirectional PSO Velocity Update Equation . . . 117
5.2.2 Dynamic Velocity Control Strategy for Non-Holonomic Robots . . . 119
5.3 Simulations and Results . . . 122
5.3.1 Obstacle course simulations . . . 123
5.4 Non-omnidirectional PSO Controller with Minimum Turning Radius . . . 125
5.5 Conclusions . . . 126

6 Marine Acoustics and Acoustic Signal Processing 130

6.1 Underwater Sound Propagation . . . 130
6.2 Marine Acoustic Sources . . . 133
6.2.1 Marine Mammal Sounds . . . 133
6.2.2 Anthropogenic Sounds . . . 134
6.2.3 Ambient Noise . . . 134
6.2.4 Signal-to-Noise Ratio Calculation . . . 135
6.3 Wavefield Correlation . . . 137
6.3.1 The Two-Hydrophone Receiver Model . . . 138
6.3.2 Ambiguities . . . 139
6.3.3 Extended Multi-Hydrophone Configurations . . . 141
6.4 Conclusion . . . 142

7 Wavefield Correlation-Enhanced Particle Swarm Optimisation 148
7.1 Amplitude-Particle Swarm Optimisation (A-PSO) . . . 148
7.1.1 Forgetting Function . . . 149
7.1.2 A-PSO Weaknesses . . . 150
7.2 Wavefield Correlation PSO . . . 151
7.2.1 Cross-Correlation Particle Swarm Optimisation (X-PSO) . . . 152
7.2.2 Bearing Particle Swarm Optimisation (B-PSO) . . . 154
7.2.3 Cross-Correlation-Bearing Particle Swarm Optimisation (XB-PSO) 157
7.3 Simulated Environment . . . 159
7.3.1 Robots . . . 159
7.3.2 Source . . . 161
7.3.3 Normalised Units and Parameter Values . . . 162
7.4 Results . . . 162
7.4.1 Generalised Results . . . 165
7.5 Conclusions . . . 169

8 Generalised Triangulation PSO and Source Entrapment/Escorting 173
8.1 Generalised Triangulation PSO . . . 174
8.1.1 Comparison with XB-PSO . . . 179
8.2 Source Entrapment/Escorting . . . 182
8.2.1 Simulations . . . 183
8.2.2 Results . . . 184
8.3 Conclusions . . . 189

9 Conclusions and Future Work 192
9.1 Conclusions . . . 192

9.2 Future Work 193

Appendices **196**

A Adapted PSO Order-1 and Order-2 Stability **197**

A.1 References 199

B Lemma 1 **200**

C Lemma 2 **201**

D Theorem 1 **203**

E Theorem 2 **204**

List of Figures

2-1 Schematic of field delimitation for Mobile Robotics, Multi-Robot Systems, and Swarm Robotics. (Dias et al. 2021). 25

2-2 Graphical representation of the optimisation process of a particle swarm in a 2-dimensional search space (i.e. two tunable parameters). The vertical dimension represents the cost of locations and the red spheres represent the particle swarm. The eventual goal of the swarm is to converge to the global minimum of the cost function. Images (a) to (c) show different stages of the optimisation process, from early to late stages (Tehrani et al. 2017). 29

2-3 Functions defining the magnitudes of virtual force vectors exerted on the robots of the swarm. In (a), the forces are exerted by the targets and they are primarily attractive forces. The region between distances do_2 and do_3 describes the ideal range that the robot needs to maintain from the target. The distance described as "predictive tracking range" signifies the limit beyond which, the target will not attract the robot. At do_1, the attractive force (positive force) becomes repulsive (negative force), to prevent collision of the robot with the target. In (b), the virtual forces are exerted by other robots in the swarm. These forces are always repulsive. Up until distance dr_1, such a force has constant repulsive effect. Beyond this distance, the magnitude of the force drops linearly and becomes 0 at distance dr_2, signifying the maximum distance at which robots can exert forces on other robots. 34

2-4 Results of the experiments carried out to assess the performance of the dPSO algorithm, using 1 robot, without obstacles. The black lines represents the route followed by the robot and the red square is the global maximum of the fitness function (brightest spot). The red points represent consecutive desired locations towards which the robot needs to move and which are calculated using dPSO. The circular shape of the robot's route is the result of its motion limitations (i.e. limited turning radius) (Hereford et al. 2007). 37

2-5 Two different taxonomies used to define swarm robotic tasks/behaviours. The taxonomy presented in (b) is an adapted version of the one presented in (a). Primary tasks such as aggregation, pattern formation, coordinated motion and task allocation appear in both taxonomies, but several new tasks are also introduced in (b) such as self-assembly, collective localisation and human-swarm interaction. These new behaviours signify the research advancements and the appearance of new questions regarding this field in recent years. 42

2-6 The ψ_α action force function is used to calculate the forces between agents of the swarm, based on the distance z between pairs of agents, in order to maintain the desired spatial separation (Olfati-Saber 2006). 44

2-7 Instances of an obstacle avoidance simulation. In (a), the swarm is initialised and the agents have not yet formed a solid group. In (b) and (c), the agents begin to come together, maintaining constant distance from neighbours and velocity consensus. In the presence of obstacles the agents separate to avoid collision. In (d) to (f), the agents rejoin together into a solid group after the obstacles have been successfully avoided. (Olfati-Saber 2006). 45

2-8 Implicit Formation Pattern Function that surrounds all targets (Zhang et al. 2018) 47

3-1 The *safe operating regions* defined by allowable values of $\omega, \hat{c}$ that guarantee order-1 and order-2 stability. 62

3-2 The phase-space graph for three different cases: (a) $\omega = 0$, (b) $0 \leq \omega < 1$ (in this specific case $\omega = 0.6$), (c) $\omega = 1$. In all three cases the system is stable. Also, in (a) (extreme case) and (b) (normal case), the system is asymptotically convergent towards the origin. 69

3-3 Example that shows the deterministic effect of $\hat{\mathbf{M}}$ (left) and the stochastic effect of $\hat{\mathbf{b}}_j[k]$ (right) on a random position of $\hat{\mathbf{z}}_j[k]$, for $\omega = 0.6$, $\hat{c} = 4$ and $\Delta t = 1$. No matter the location of $\hat{\mathbf{z}}_j[k]$, the point $\hat{\mathbf{M}}\hat{\mathbf{z}}_j[k]$ will always be located closer to the origin, lying on a_1. The point $\hat{\mathbf{z}}_j[k+1]$ will always be located in-between the lines a_2 and a_3. The vector $\hat{\mathbf{b}}_j[k]$ is always parallel to the hatching lines of the shaded-hatched region, which have gradient $\frac{1}{\Delta t}$. The shaded-hatched region represents all possible states $\hat{\mathbf{z}}_\mathbf{j}[k+1]$. For this system, $A_j^+ = 4\,\mathrm{m/s^2}$, $A_j^- = 8\,\mathrm{m/s^2}$ and $U_j = 10\,\mathrm{m/s}$. 71

3-4 The maximum observed velocities at each second, for swarms of 10, 100 and 1000 particles (in order from darker green to lighter green regions). The blue solid line represents the value of the desired maximum velocity U and the red dashed line represents the value of the Acceleration-Time product ($A^+ \times \Delta t$. The yellow solid line is the velocity of a random particle. In (a) and (d), the simulations have the same sensitivity factor $\beta = 1$ and showcase similar maximum observed velocities. Similarly, in (b) and (c), the sensitivity factor is $\beta = 0.2$ and the simulations showcase similar maximum observed velocities. 76

3-5 The maximum observed velocities at each second, for swarms of 10, 100 and 1000 particles (in order from darker green to lighter green regions). The blue solid line represents the value of the desired maximum velocity U and the red dashed line represents the value of the Acceleration-Time product ($A^+ \times \Delta t$. The yellow solid line is the velocity of a random particle. In (a), the desired maximum velocity U decreases at 34 s and increases again at 67 s varying β from low to high to low. In (b), both the desired maximum velocity U and acceleration A^+ decrease at 34 s and increase again at 67 s varying β from high to low to high. In (c), both the desired maximum velocity U and acceleration A^+ decrease at 34 s and increase again at 67 s while β remains high at all times. 78

4-1 The obstacle map used in both MATLAB and Gazebo simulations. The blue square on the left shows the starting area where robots are initialised and the red square on the right shows the position of the source. The obstacles become denser the closer to the source. 96

4-2 (a) shows the real Robotnik Summit XL Steel platform (Robotnik Automation S.L.L. n.d.), while (b) shows a simulated modified model of the platform used in some of the following simulations. The robots are equipped with mecanum wheels for holonomic motion and a contact sensor (green link) to detect collisions. 97

4-3 The three possible cases that are covered by the calibration strategy during DVC. The green circle represents the robot and the red crossed circles are the obstacles. The dotted lines show how the six sensing regions are separated and the dashed circle represents the maximum detection range of the robot. If no obstacle is present, as in (a), $c_1 + c_2$ (blue region) is maximised and $c_3 = 0$. If one obstacle is present, as in (b), $c_1 + c_2$ is limited by the distance to the obstacle; c_3 (orange region) is increased to cover the extra potential for movement. If two or more obstacles are present, as in (c), $c_1 + c_2$ is limited by the distance to the closest obstacle and c_3 (orange region) is increased to cover the extra potential for movement but it is also limited by the distance to the second closest obstacle. 99

4-4 Image of the Gazebo environment during the operation of the obstacle course simulations. The global minimum of the fitness function (source) is assumed to be located on the right side of the image outside the obstacle course (the source is not represented by an actual object in order to avoid collisions with the robots). The grey cylinders are the obstacles while the green and red circles are the robots. The green robots are currently operational while the red robot has collided with an obstacle. 100

4-5 Graph of the flow of information between ROS nodes (circles) and ROS topics (rectangles) for a swarm of 2 robots, during the obstacle course simulations. For the operation of the PSO algorithms, each robot maintains four nodes (PSO_control, check_pbest, mecanum_control and produce_avoid_vectors) and four local topics (odom, avoid_agents, pso_vel and cmd_vel). There also exist two global topics that can be accessed by all robots (all_pos which contains the current positions of all robots and p_best which contains the current personal best locations of all robots). 101

4-6 Median CoM fitness over time results for different cases. The dotted lines represent the obstacle layers of the obstacle course. 103

4-7 The expected number of collided vs operational robots at the end of the median simulation for different cases. 104

5-1 Visualisation of the PSO particles for the 2D and the 3D case. The particles tend to gather on the global coordinate axes, forming cross-shaped patterns. This is understood to be the result of the decoupling of the velocity components, along with the use of global coordinate axes. (Spears et al. 2010) 111

5-2 Schematics that show the separation of the velocity components for a robot-centred frame of reference, where the green circle is the robot and the yellow arrow indicates its orientation. In (a), the velocity components v_1 and v_2 represent the longitudinal and lateral linear velocities of the robot respectively, while in (b), the velocity components v and w represent the linear and angular velocities of the robot respectively. . . . 113

5-3 Schematics that explain the DVC tuning strategy for the control of non-holonomic vehicles. The vehicle is described by the centred circle, where the arrow describes its orientation. The numbered areas represent the sensing regions of the vehicle and the circular dashed line represents its maximum sensing range. The red circles represent obstacles. In (a), the sensing regions are numbered. In (b), one obstacle is detected on the front left of the robot. This will cause a decrease in the linear velocity (represented by a decrease in the green region) and it will cause the robot to rotate rightwards. Furthermore, the linear velocity of the robot will be limited. In (c), There exist two obstacles, one on the left and one on the right, at an equal distance from the robot. This will limit the linear velocity of the robot but it will not cause any rotation. 120

5-4 A simulated swarm of 20 robots (green circles) that are controlled by the non-omnidirectional PSO controller. The red lines indicate the past positions of each robot in the last 15 seconds. The robots never stop moving while no collisions occur between them. 123

5-5 Median CoM fitness over time results for the non-omnidirectional and the omnidirectional PSO controllers. The transparent areas also show the 5^{th} and 95^{th} percentile CoM fitnesses of each controller. The dotted lines represent the obstacle layers of the obstacle course. 124

6-1 Schematics that describe the two main types of signal propagation, using a ship-mounted SONAR as an acoustic source example. 132

6-2 Composite of ambient-noise spectra, summarizing results and conclusions about spectrum shape, level, and probable sources of ambient noise between 1 Hz and 100kHz (Wenz 1962). 136

6-3 Schematic that describes how an incoming signal emitted by an external wavefield source (ship) is received by a pair of hydrophones (black circles). In this case, the signal is first received by the right-most hydrophone and after time τ_{lag}, it is received by the left-most one. The angle α is the AOA of the signal (i.e. direction towards the source). 138

6-4 Schematics that describe the two problems of directional ambiguity, where the black circles represent the pair of hydrophones. In (a), the angle α is the calculated angle of arrival of the signal. The solid and dashed green lines are the leftward and rightward candidate directions towards the source respectively. In (b), the green (middle) signal represents the signal emitted by the source, while the red and blue (side) signals represent ambiguities caused by the hydrophone separation. The corresponding red, green and blue phase profiles show how all three signals can result in the same phase difference for the two hydrophones (arrows on the phase-distance graph). 140

6-5 Normalised polar sensitivity pattern for hydrophone arrays with different number of elements. The hydrophone arrays are placed vertically (i.e. the -90^o to 90^o line) and consecutive hydrophones are separated by a distance of $\lambda/2$, where λ is the wavelength of the received signal. The figures were obtained using the MATLAB Phased Array System Toolbox. 142

6-6 Simulated performance results for two different hydrophone configurations, for the task of source localisation. In (a) and (b), even though a third non-colinear hydrophone is used, its improper placement does not fully resolve ambiguities. In (c) and (d), a fourth hydrophone is used and the hydrophones are more uniformly distributed and therefore ambiguities are fully resolved. The simulated signal used was broadband with frequencies in the range 1 kHz to 10 kHz, the signal propagation speed was 1500 m/s and the SNR of the hydrophone signal readings was 10 dB (Wang et al. 2019). 143

7-1 A swarm of robots (white rectangles) that uses B-PSO to estimate the location of an acoustic source (ship). Solid rays are source rays and dashed rays are ambiguous rays. The numbers next to each ray intersection (circles) represent the type of intersection. The red circles indicate the intersections that would likely be selected by at least one robot (i.e. furthest intersection from the robot). In this way, selected intersections are more likely to be intersections of source rays. 154

7-2 Distances of CoM to source at different initial SNR values (SNR_0) for the tested algorithms. The solid lines represent the median distance from source over 100 simulations and the transparent areas the 90% percentile range. 163

7-3 Graphical representation of the routes followed by the CoM of 100 swarms for each method. The image at the bottom plane of each graph shows contours of the number of swarms (CoM) that passed from different locations. The red, green and blue lines represent the routes followed by three randomly selected swarms with respect to time. The corresponding dashed lines represent the 2D projections of these routes on the Timesteps vs Spatial-steps vertical plane. 164

7-4 Normalised time needed for the CoM of a swarm to reach convergence (distance d_c from the source) for each algorithm, for different values of Q and $^{D}/_{\lambda_c}$. Each point represents the median performance over 100 swarms. The red lines represent the locations where the Q-factor is 1 and 10. The vertical blue solid line represents the simulations of Figure 7-2. The vertical black dashed and cyan dashed-dotted lines represent the Q-factors of the sound generated by two electrically-propelled UUVs, REMUS-100 and Odyssey IIb respectively (Gebbie et al. 2012, Zimmerman et al. 2005, Holmes et al. 2010). Note that although a D/λ_c axis is included for A-PSO, it does not affect the position of the hydrophone, since only one is used for that algorithm, located in the middle of the robot. 167

7-5 Normalised time needed for the CoM of a swarm to reach convergence (distance d_c from the source) for each algorithm, for different values of U and Q. Each point represents the median performance over 100 swarms. The red lines represent the locations where the Q-factor is 1 and 10. The vertical blue solid line represents the simulations of Figure 7-2. The vertical black dashed and cyan dashed-dotted lines represent the Q-factors of the sound generated by two electrically-propelled UUVs, REMUS-100 and Odyssey IIb respectively (Gebbie et al. 2012, Zimmerman et al. 2005, Holmes et al. 2010). 168

8-1 Schematics that explain the operation of Generalised Triangulation PSO. The red spheres represent the locations of robots and the green sphere represents the location of the source. The height of each robot's position-intensity (black spheres) represents the average intensity of the sensor readings of the corresponding robot. The blue triangle is the position-intensity plane and the red arrow represents its normal. The dashed red arrow is the projection of the normal on the xy-plane and it is collinear with the source. In (a), the z-component of the normal is negative and therefore its projection points towards the source. In (b), the z-component of the normal is positive and therefore its projection points away from the source. 175

8-2 Percentile Distances of CoM to source at different SNR_0 for the tested algorithms. The solid lines represent the median route over 100 simulations, the 50% transparent areas represent the 50% percentile range and the 90% transparent areas the 90% percentile range. 181

8-3 Instances of the simulations carried out for the task of source entrapment/escorting. In the simulations, a swarm of robots (white rectangles) is tasked to maintain a minimum distance r, represented by the red ring, from the source (red target). In (a), the source is stationary while in (b), the source is moving towards the right with normalised velocity $v_n = 0.25$. In both simulations, the number of robots in the swarm is 40 and the power spectral density of the emitted signal at source level is 120 dB re 1µPa^2/Hz. The blue lines represent the path of each robot in the last 15 timesteps. 184

8-4 Probabilistic distribution of the swarm relative to a stationary source (red dot). The source emits a signal with constant power spectral density PSD_s that the robots can detect. The swarm consists of N robots that are required to maintain a specific distance r (red ring) from the source. The shaded regions represent the probability that a robot will be found in a specific location at any given point. The black outline represents the locations where, there exists a 0.1% probability that a robot will be found there at any point. 185

8-5 Probabilistic distribution of the swarm relative to a source (red dot). The source is moving with normalised velocity v_n towards the right and it emits a signal with constant power spectral density PSD_s that the robots can detect. The robots are required to maintain a specific distance r (red ring) from the source. The shaded regions represent the probability that a robot will be found in a specific location at any given point. The black outline represents the locations where, there exists a 1% probability that a robot will be found there. 187

9-1 Flow chart that explains which of algorithms proposed in this thesis could be used for the control of a robotic swarm, based on the motion characteristic of the robots in the swarm and the source that needs to be localised. 194

List of Tables

4.1 Table of values used for different parameters 95

5.1 Table of values used for different parameters of the two PSO controllers for the obstacle course simulations. 124

7.1 Selected parameter values to approximate a marine source localisation scenario 160

8.1 Selected parameter values to approximate a marine source localisation scenario 180

Chapter 1

Introduction

1.1 Thesis Motivation

The oceans cover more than 70% of the earth's surface and they contain around 97% percent of its resources in water. They also comprise over 90% of the planet's living space and they play a significant role in the regulation of the earth's environmental balance, either by the massive contribution of oxygen into the planet's atmosphere through phytoplankton photosynthesis or the large amounts of evaporated water that is then transported to land through precipitation (Jonasdottir 2016). Throughout history, humans have made use of the sea as a means for sustenance, transport, commerce, growth, and inspiration.

With the advancement of society and technology, the increasing number of human activities in oceanic and marine environments has gradually affected their natural balance, through the introduction of noise and chemical pollution, which can eventually lead to catastrophic consequences (Hildebrand 2005). Significant research is therefore currently focused on understanding the impact that human activities have on oceanic environments and marine life, in order to allow proper regulations and measures to be introduced. Despite the commonly acknowledged importance of oceans, more than 80% of them is yet to be observed, mapped or explored. This number does not only include the underwater landscape, but also life below the ocean surface. To properly understand how anthropogenic pollution affects marine life, it is therefore important to also study the different marine environments and species themselves.

Traditional marine environment monitoring systems employ a single oceanographic research vessel, equipped with all the necessary sensors for the acquisition of data (Wüst

1964). This method of study of marine environments is expensive, time-consuming and unable to study efficiently the large areas that need to be explored. Furthermore, its measurements often have very low resolution in both time and space (Xu et al. 2014). To address this, recent research has focused on the development of wireless sensor networks, which can be low-cost, semi-disposable and enable the distribution of sensors over large areas for simultaneous data collection and higher resolution measurements. Such networks typically consist of a large number of dedicated sensor nodes capable of sensing the environment around them and limited processing of the collected data, before they are transmitted to a central location (Xu et al. 2014). In the last decade, wireless sensor networks have been successfully used in applications such as water monitoring (Jiang et al. 2009, Pérez et al. 2011), animal behaviour monitoring (Cedeño-Antunez et al. 2019), disaster prevention (Iacono et al. 2010) etc.

Despite their advantages, wireless sensor networks also come with several disadvantages. Namely, they require higher water resistance and stronger robustness to ensure survival and proper operation in aggressive and dangerous environments, higher energy consumption due to communication and data transmission and they are susceptible to measurement disturbances that can be caused by the interference from antennas used for communication. Additionally, static networks (e.g. ones that employ buoy devices as sensor nodes) can be negatively affected by the unintentional movement of the sensors or they may be incapable of studying moving targets (e.g. marine species)(Xu et al. 2014).

To address this last disadvantage, the research community has focused on the introduction of autonomous movement capabilities into wireless sensor networks, which can enable the individual nodes to be used for the localisation of marine species and their continuous monitoring. A research field that offers methodologies particularly suitable for the decentralised control of a large number of low-cost, semi-disposable nodes is that of swarm robotics. The emergent swarm behaviours of robotic swarms are inherently characterised by scalability, adaptability and robustness, thereby satisfying several of the requirements of wireless sensor networks (Couceiro et al. 2013, Shin & Lee 2020).

Swarm robotics is a relatively new field of study. What differentiates it from other cooperative robotic systems is that cooperative behaviour is not explicitly programmed in the control algorithms of the robots. Instead, it emerges from the decisions of individual robots, thereby allowing the control of an arbitrarily large number of robots without increasing computational complexity. Additionally, no single robot is crucial for the correct operation of a swarm. This makes robotic swarms particularly suitable for use in dangerous and unexplored environments where the loss of several robots may

be unavoidable. While the field of swarm robotics has seen significant advancement in recent years, standardisation of its methodologies is a milestone that has not yet been achieved (Nedjah & Junior 2019). The inherent complexity of swarm behaviours makes swarm control algorithms difficult to be implemented. Additionally, the need to acquire and maintain a large number of individual robots makes the real-world testing of robotic swarm systems difficult. For this reason, validation of generalisable swarm control algorithms is usually limited to the use of simulations, while the small number of tested real-world swarm robotic systems are typically application specific (Hamann 2018). Therefore, to open the way towards the implementation of real-world autonomous marine swarms, it is first desirable to show how swarm control algorithms could achieve their proper control.

This thesis will focus on the study and development of algorithms for the control of marine robotic swarms with the task of underwater acoustic source localisation and monitoring. Marine acoustic sources that can be localised and monitored using such a system mainly include biological sources (e.g. marine mammals) and anthropogenic sources (e.g. transportation vessels, unmanned underwater vehicles etc.). The thesis will aim for the development of generalised algorithms in order to contribute towards the standardisation of the field of swarm robotics with specific focus on the tasks of source localisation and monitoring.

1.2 Covid-19 Pandemic Impact

The project described in this thesis was affected by the 2020 Covid-19 pandemic lockdown. The presented algorithms and simulation results were planned to be validated using real-world experiments that would take place during the 2020-2021 period and which were ultimately not carried out. Instead, the MATLAB simulations were validated using Gazebo, a realistic robot simulation software that benefits from a detailed physics engine.

1.3 Contributions

The contributions of this thesis to the fields of swarm and marine robotics are:

1. A novel control algorithm is introduced by modifying a prevalent swarm intelligence algorithm (particle swarm optimisation), allowing low-level motion control of robotic swarms. The algorithm offers the following advantages that are currently not shared by other swarm intelligence algorithms:

- It enables direct control of the motion of the robots by taking into account kinematic and dynamic constraints posed by the robot.
- It allows consideration of the refresh rate of each robot's controller, resulting in the proper synchronisation of the controller with the physical robotic platform.
- It enables the merging of a large number of swarm robotic off-the-shelf algorithms and techniques, meant to address different swarm robotic tasks, while maintaining their scalability, adaptability and robustness - a milestone that has not yet been reached in the swarm robotic literature.
- Its inherent functionality allows it to be adapted for use in a variety of swarm robotic scenarios, beyond source localisation and monitoring.

These capabilities of the algorithm are demonstrated in Chapters 3 to 5 using MATLAB and Gazebo simulations for robots with both omnidirectional and non-omnidirectional motion.

2. Through further modifications of particle swarm optimisation, three additional algorithms are introduced in Chapter 7 that enhance its source localisation capabilities using wavefield correlation techniques currently used in multi-hydrophone array systems. The proposed algorithms achieve improved localisation range and faster and more consistent convergence towards the source. The algorithms are tested using MATLAB simulations and the results are generalised for use in acoustic source localisation of various signal characteristics.
3. Finally, Chapter 8 demonstrates how the algorithms introduced in the rest of the thesis can be combined to allow interaction with the source. In this way, the swarm can avoid, encircle and follow the source, allowing its continuous monitoring. This capability, demonstrated through MATLAB simulations for both stationary and moving sources, is not offered by current source localisation algorithms used in swarm robotics and it is crucial for the implementation of a real-world system capable of source monitoring.

Some of the work presented in this thesis is based on the following publications:

- Rossides G, Metcalfe B, Hunter A. Particle Swarm Optimization − An Adaptation for the Control of Robotic Swarms. *Robotics*, 10(2), 2021.
- Rossides G, Hunter A, Metcalfe B. Source Localisation using Wavefield Correlation enhanced Particle Swarm Optimisation. *Robotics*, 11(2), 2022.

References

Cedeño-Antunez, U., Carvajal-Gamez, B. E. & Pallares-Calvo, A. E. (2019), Wireless System Based in Cellular Network for Monitoring Marine Mammals at Mexican Coast, *in* '2019 IEEE 8th International Workshop on Advances in Sensors and Interfaces (IWASI)', pp. 251–254.

Couceiro, M. S., Rocha, R. P. & Ferreira, N. M. F. (2013), 'A PSO multi-robot exploration approach over unreliable MANETs', *Advanced Robotics* **27**(16), 1221–1234.
URL: *https://doi.org/10.1080/01691864.2013.819605*

Hamann, H. (2018), *Introduction to Swarm Robotics*, Springer International Publishing, Cham, pp. 1–32.
URL: *https://doi.org/10.1007/978-3-319-74528-2_1*

Hildebrand, J. (2005), Impacts of Anthropogenic Sound.

Iacono, M., Romano, E. & Marrone, S. (2010), Adaptive monitoring of marine disasters with intelligent mobile sensor networks, *in* '2010 IEEE Workshop on Environmental Energy and Structural Monitoring Systems', pp. 38–45.

Jiang, P., Xia, H., He, Z. & Wang, Z. (2009), 'Design of a water environment monitoring system based on wireless sensor networks.', *Sensors (Basel, Switzerland)* **9**(8), 6411–6434.

Jonasdottir, S. (2016), State of the earth´s oceans. Sustain Abstract S-3; Sustain-ATV Conference 2016 : Creating Technology for a Sustainable Society ; Conference date: 30-11-2016 Through 30-11-2016.
URL: *http://www.sustain.dtu.dk/about/sustain-2016*

Nedjah, N. & Junior, L. S. (2019), 'Review of methodologies and tasks in swarm robotics towards standardization', *Swarm and Evolutionary Computation* **50**, 100565.

Pérez, C. A., Jiménez, M., Soto, F., Torres, R., López, J. A. & Iborra, A. (2011), A system for monitoring marine environments based on Wireless Sensor Networks, *in* 'OCEANS 2011 IEEE - Spain', pp. 1–6.

Shin, C. & Lee, M. (2020), 'Swarm-Intelligence-Centric Routing Algorithm for Wireless Sensor Networks', *Sensors* **20**(18).
URL: *https://www.mdpi.com/1424-8220/20/18/5164*

Wüst, G. (1964), 'The major deep-sea expeditions and research vessels 1873-

1960. A contribution to the history of oceanography', *Progress in Oceanography* **2**, 1,3,IN1,7,IN6,11,IN9,15,IN13,19–1,6,IN4,10,IN7,14,.

Xu, G., Shen, W. & Wang, X. (2014), 'Applications of Wireless Sensor Networks in Marine Environment Monitoring: A Survey', *Sensors* **14**(9), 16932–16954.
URL: *https://www.mdpi.com/1424-8220/14/9/16932*

Chapter 2

Introduction to Swarm Robotics

Swarm robotics is a part of the umbrella of cooperative robotics and Multi-Robot Systems (MRS), as shown in Figure 2-1. The main difference between swarm robotics and other MRS approaches is that it aims to achieve control of large numbers of robots (tens to thousands) that can collaboratively achieve a specific goal. More specifically, it is defined by Şahin (2005) as:

> "The study of how a large number of relatively simple, physically embodied agents can be designed such that a desired collective behaviour emerges from the local interactions among agents and between the agents and the environment."

From here, several characteristics emerge that can be used to differentiate a robotic swarm from other MRS (Şahin 2005, Senanayake et al. 2016):

- The robots that consist the swarm should be autonomous agents that can move and sense the real-world. Alternatively, particles that exist in a simulation and are controlled by a specific algorithm could also be considered a swarm as long as the algorithm treats them as individual particles that move using information that they have obtained on their own or have been shared by other particles in the swarm.

- The algorithm that controls the robots must be designed to control large swarms. Note that this does not prohibit the use of a small number of robots in a swarm, as long as the emergent behaviour is scalable to large numbers. This is often a source of confusion, as people can be mislead to think that robotic swarms must necessarily be large. In fact, the phrase "graceful performance degradation" can be used to describe how the performance of robotic swarms degrades gracefully

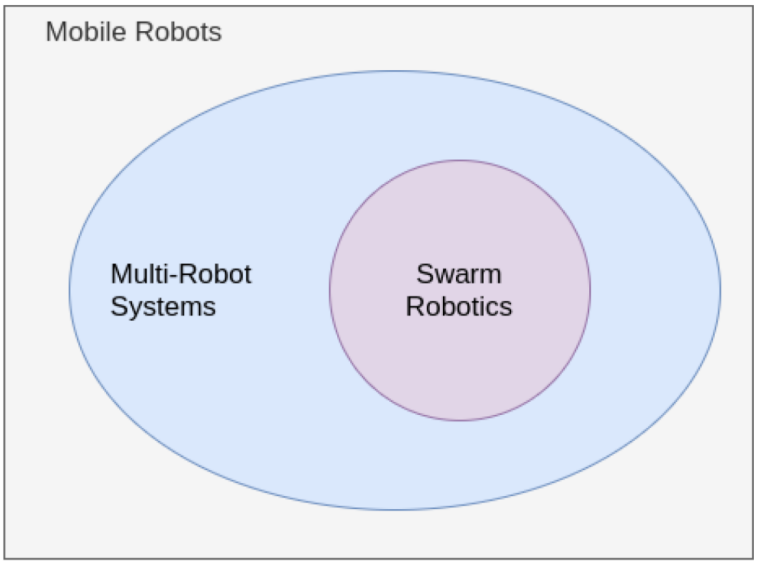


Figure 2-1: Schematic of field delimitation for Mobile Robotics, Multi-Robot Systems, and Swarm Robotics. (Dias et al. 2021).

with a smaller number of robots, in contrast to other cooperative robotic schemes that may completely stop operating as soon as a single robot is lost (Portugal & Rocha 2016).

- Even though heterogeneous swarms can exist (i.e. swarms that consist of different types of robots), each heterogeneous swarm must consist of homogeneous groups (i.e. groups that consist of the same type of robots), where each homogeneous group must have a fair number of robots to ensure scalability. In other words, a heterogeneous swarm must consist of at least one homogeneous sub-swarm.

- The individual robots must be relatively incapable of achieving the overall goal on their own. The goal should only be achievable through the emergent collaborative behaviour of the whole swarm.

- The robots should make their decisions based on local information collected by themselves or communicated by other robots. This is an important characteristic that aims to simplify the control algorithms and promote scalability. As the goals of individual swarm robotic systems become more complicated, the practitioner may be tempted to achieve them by supplying the individual robots of the swarm with more information. This bears the risk of making the control algorithm application-specific or even non-scalable. Instead, the abilities of the individual robots should be kept constrained and if necessary, a larger number of

robots should be introduced.

The term swarm intelligence was first proposed by Beni & Wang (1993) and it is considered to be a subset of artificial intelligence. When it comes to robotics, it refers to the algorithms used for the control of robotics swarms (Beni 2005, Sharkey 2007), but it has been also successfully used in a number of other applications, including numerical optimisation. In engineering disciplines, swarm engineering (Kazadi 2000, Winfield et al. 2005) refers to the discipline that aims to combine dependable systems and swarm intelligence.

Systems that employ swarm intelligence share the following three main characteristics (Bayindir & Sahin 2007, Senanayake et al. 2016):

- **Scalability**: The ability of an algorithm to control the behaviour of an arbitrary number of individuals. The number of individuals should not affect the performance of the algorithm considerably. In swarm robotics, this ensures that the same algorithm can be used on a large swarm of 1000 robots and a small swarm of 10 robots, with the overall emergent behaviour being fairly similar.

- **Adaptability**: The ability of the system to adapt to different environmental conditions. In swarm robotics, this is achieved through the constraint that is applied on the robots to only be able to sense their local environment. As long as each robot is capable of reacting correctly to different local factors, it does not matter how the overall global environmental factors change over time - the swarm will be able to adapt correctly.

- **Robustness**: The ability of a system to continue operating correctly, even in the presence of partial failures or imperfect environmental conditions. Robustness is a direct by-product and an extension of scalability. In swarm robotics for example, even if a number of individual robots break down during operation, the performance of the swarm will degrade gracefully, due to its inherent scalability, making the overall system also more robust. On the other hand, robustness also adds the condition that if a small number of robots end up supplying the swarm with bad information (e.g. due to faulty sensor readings), the swarm should still be able to achieve its goal.

Based on these characteristics of swarm intelligence, several other characteristics can be derived that are inherent to swarm robotics systems, such as the use of simple, low-cost, semi-disposable robots and derivative-free optimisation control (i.e. optimisation of the swarm behaviour without the need for complex higher-order information about

the environment). In turn, these characteristics of swarm robotics systems make them perfect candidates for applications that require operation in large, unexplored and possibly dangerous areas, where the landscape is unknown, various threats can exist and the loss of individual robots may be unavoidable.

The rest of this chapter will aim to present in depth, the algorithms and methodologies that constitute the fields of swarm intelligence and swarm robotics. Example applications will also be presented for each algorithm. Section 2.1 will present existing swarm intelligence algorithms, Section 2.2 will describe a number of swarm robotic tasks and behaviours, relevant to the overall goal of this thesis. In the end, Section 2.3 will discuss gaps in the literature and opportunities that can be identified from the current state of the field.

2.1 Swarm Intelligence Algorithms

Swarm Intelligence Algorithms (SIAs) were initially almost exclusively bio-inspired. Nature has provided a plethora of examples of cooperative and swarm behaviour and researchers have drawn inspiration from these for the development of early SIAs. Some examples of biological sources of cooperative and swarm behaviour are ants, bees, fireflies, glow-worms, bats, monkeys, lions and wolves (Chakraborty & Kar 2017).

With the passing of time and the advancement of swarm intelligence studies, several key concepts were extracted from the bio-inspired algorithms and were used for the creation of new SIAs that could no-more be characterised as bio-inspired (Senanayake et al. 2016). Such concepts include:

- stigmergy (Grassé 1959, Theraulaz & Bonabeau 1999), where robots exchange information through the use of artificial pheromones that they leave behind at locations from which they have passed.
- potential fields (Khatib 1985), which make use of global, virtual force vector fields for the coordination and movement control of robots.
- physics-based laws to compute local virtual forces (Spears et al. 2004), where instead of global virtual force fields, each robot computes the virtual forces on its own, based on local information that it has collected.
- flocking (Reynolds 1987), which concerns itself with the coherent coordination and movement control of the swarm, so that it can travel efficiently over long distances.

- population-based stochastic optimisation (Nayak et al. 2019), where a large number of robots are used to identify the global extremum of a fitness/cost function.

The rest of this section will present a number of the most popular SIAs and control techniques currently provided in the literature (both bio-inspired and non-bio-inspired). Since the overall goal of this thesis is the development of a system used for source localisation and area searching, the following presentation of SIAs will focus on the ones that are mostly related to these tasks. Applications of these algorithms will also be presented.

2.1.1 Particle Swarm Optimisation

One of the most popular and widely-used SIA is Particle Swarm Optimisation (PSO) (Kennedy & Eberhart 1995). It is a population-based stochastic optimisation algorithm that was inspired from the behavioural model of Reynolds (1987), created to describe the collective behaviour of bird flocks and fish schools. PSO was created with simplicity in mind and for use in multi-dimensional numerical optimisation tasks, where a population of virtual particles explore the problem space, evaluating the fitness/cost of different locations (corresponding to tunable parameter values). Its inherent scalability and bio-inspired nature eventually motivated its proposal for use for the control of robotic swarms (Hereford et al. 2007, Pugh & Martinoli 2007). This is achieved by equating the behaviours of real-world robots of a swarm to the behaviours of PSO particles.

The fundamental aim of PSO is to identify the location inside a multi-dimensional space that minimises a cost function (or maximises a fitness function) through the use of a swarm of virtual particles. At each timestep, every particle calculates the cost of its current location. Each particle remembers its past location that resulted in the lowest cost (personal best location). The particles share their personal best locations with the rest of the swarm and the one with the lowest cost is selected (global best location).

The motion of each particle is described by the velocity update equation

$$\mathbf{u}^i[k+1] = \omega \mathbf{u}^i[k] + c_1 \mathbf{r_1} \circ (\mathbf{y}^i[k] - \mathbf{x}^i[k]) + c_2 \mathbf{r_2} \circ (\mathbf{y_g}[k] - \mathbf{x}^i[k]), \tag{2.1}$$

and the position update equation

$$\mathbf{x}^i[k+1] = \mathbf{x}^i[k] + \mathbf{u}^i[k+1], \tag{2.2}$$

where $\mathbf{u}^i[k]$ and $\mathbf{x}^i[k]$ are the velocity and position of particle i at timestep k. The $\circ$ operator denotes element-wise multiplication and the vectors $\mathbf{r_1}$ and $\mathbf{r_2}$ are vectors of

random components in the range [0,1). The locations $\mathbf{y}^i$ and $\mathbf{y_g}$ are the personal best location for particle i and the global best location respectively. The PSO parameters ω, c_1 and c_2 are known as the inertia weight, the cognitive coefficient and the social coefficient respectively (Cleghorn & Engelbrecht 2018). Alternatively, c_1 and c_2 are known as the accelerating coefficients and their corresponding terms in (2.1) are known as the accelerating terms.

After initialising the particle swarm inside a multi-dimensional environment, the particles will make use of their own measured location costs as well as the ones communicated by other particles to converge towards locations of low cost. The purpose of the algorithm is for the swarm to eventually converge on the global minimum of the cost function, as shown in Figure 2-2. Swarm robotic applications of PSO are typically source localisation applications, where the fitness/cost function is defined with respect to properties of the signal emitted by the source (e.g. signal intensity). In this way, the global extremum of the fitness/cost function (e.g. location of maximum signal intensity) is the location of the source.

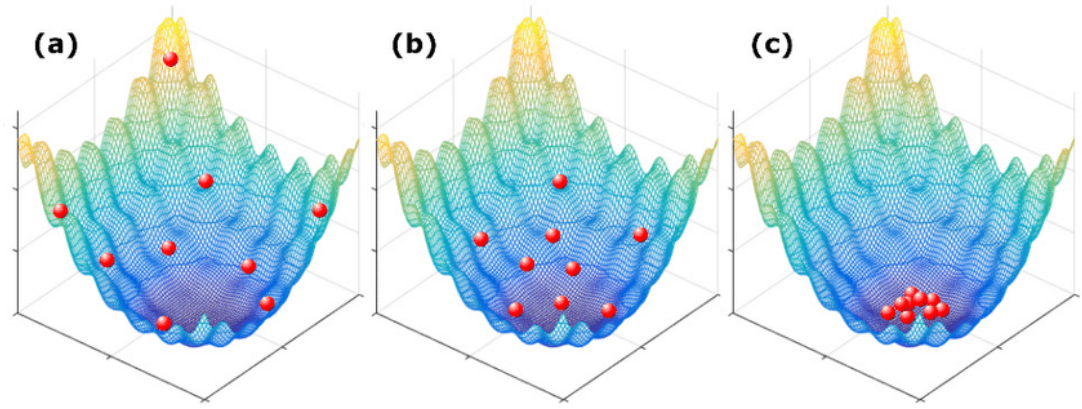


Figure 2-2: Graphical representation of the optimisation process of a particle swarm in a 2-dimensional search space (i.e. two tunable parameters). The vertical dimension represents the cost of locations and the red spheres represent the particle swarm. The eventual goal of the swarm is to converge to the global minimum of the cost function. Images (a) to (c) show different stages of the optimisation process, from early to late stages (Tehrani et al. 2017).

By adjusting the value of the inertia weight ω, the convergence rate of the swarm is controlled (Juneja & Nagar 2016), such that, small values of ω will result in steady and rapid convergence towards the currently known minima of the cost function, while large values of ω will cause the particles to overshoot as they pass from the known maxima, thereby promoting exploration but slowing the convergence of the swarm.

The values of the cognitive coefficient c_1 and the social coefficient c_2 can be adjusted to further control the exploration/exploitation tendencies of the swarm (Juneja & Nagar 2016). When $c_1 \gg c_2$, the particles will prioritise movement towards their own personal best locations $\mathbf{y}$, with the aim to spend more time exploring the different areas of the simulated environment for the identification of minima. Conversely, when $c_1 \ll c_2$, the particles will prioritise movement towards the current global best location $\mathbf{y_g}$. This will cause the whole swarm to converge quickly towards a single location, exploring the area in its way. After convergence to a location has been achieved, further exploration is very limited - this is known as the Diversity Loss problem (Blackwell 2007).

PSO is widely used and there have been numerous studies about its different characteristics, behaviours and variations. Despite its popularity, to date it has not been possible to prove mathematically that PSO will eventually converge to the global minimum of its cost function. In recent stability analyses (Ozcan & Mohan 1999, Trelea 2003, Liu 2014, Bonyadi & Michalewicz 2016, Cleghorn & Engelbrecht 2018), it has been possible to show that PSO particles will always remain stable (i.e. will not diverge uncontrollably) when certain PSO parameter values are used, but all of these analyses use simplifying assumptions.

Most of the PSO variants are specifically created for use in numerical optimisation problems. That said, some of these variants can be applied to swarm robotic applications. Some of the most important variants (Eberhart & Shi 2001) replace the global best location $\mathbf{y_g}$ of the original PSO with a neighbourhood best location $\mathbf{y_n}$, which is calculated using the personal best locations $\mathbf{y}$ of surrounding robots instead of all the robots in the swarm. This can be especially useful in swarm robotics, since physical robots typically have a limited communication range and they may only be able to communicate with a limited number of other robots in the swarm.

Other useful variants are concerned with the use of dynamic cost functions (Carlisle & Dozier 2000, Fernandez-Marquez & Arcos 2009). The original PSO requires an immutable environment (i.e. when personal best location $\mathbf{y}^i$ is selected by robot i, it is assumed that the cost of that location does not change over time). This can be a problem when noisy or dynamic cost functions are used (i.e. the cost of a location changes over time) - this is known as the Outdated Memory problem of PSO (Blackwell 2007). In the real-world, cost functions are expected to be both noisy (e.g. noise in the signal readings of a robot) and dynamic (e.g. if the source moves, the cost at certain locations will change). Therefore, PSO variants that deal with dynamic cost function can be especially useful in swarm robotics (Zhang et al. 2019, 2020).

2.1.2 Ant Colony Optimisation

Another popular swarm intelligence algorithm is Ant Colony Optimisation (ACO) (Dorigo et al. 2006). ACO was inspired from the foraging behaviour of ants. In nature, ants employ the concept of stigmergy to minimise the travelling distance to sources of food. As the individual ants roam randomly in search of food, they leave trails of pheromones that fade away over time. When other ants detect the trail of pheromones, they follow it, enhancing it with their own pheromones. Over time, the ants tend to follow and enhance shorter trails, while the longer ones are left to fade away. In this way, the shortest route to the food source is identified. The algorithm has been successfully used to tackle a variety of problems including data mining (Abraham & Ramos 2003), the travelling salesman problem (Gambardella & Dorigo 1996) and the vehicle routing problem (Bullnheimer et al. 1999).

2.1.3 Artificial Bee Colony Optimisation

Artificial Bee Colony Optimisation (ABC), is a numerical optimisation algorithm proposed by Karaboga (2005), which aims to emulate the foraging behaviour of honeybees. In nature, honeybees employ the waggle-dance (Biesmeijer & Seeley 2005) to communicate the location of discovered food sources to other members of the colony. In ABC, the individuals in the swarm are separated into three categories: the scouts, the onlookers and the employed bees. The scouts randomly search the environment for food sources (locations of high fitness). As soon as a scout discovers a source of food, it becomes an employed bee, executing a virtual waggle-dance to communicate its location to other bees. At that point, a number of onlookers that were waiting at the hive are assigned to the source and begin exploiting it. If the source runs out of food (i.e. fitness of location decreases), the onlookers return to the hive and the employed bee associated with the source becomes a scout again. ABC has several important advantages over other algorithms. Due to the way that it operates, exploration and exploitation happen simultaneously and never stop. Furthermore, since it considers that sources of food can eventually run out, it can be used for mutable problems and environments.

2.1.4 Glowwarm Swarm Optimisation

Glowworm Swarm Optimisation (GSO) (Krishnanand & Ghose 2009) is a bio-inspired population-based stochastic optimisation algorithm designed to emulate the behaviour of glowworms. In GSO, each individual glowworm emits a virtual luminescence quantity called 'luciferin', which increases or decreases gradually as the glowworm passes from locations of high fitness or low fitness. At each timestep, each glowworm chooses

probabilistically another glowworm and moves towards it and glowworms with higher luciferin have higher probability to be selected. The algorithm works in two stages for each timestep k. In the first stage, the luciferin $l^i[k]$ of glowworm i is updated using the luciferin update rule

$$l^i[k+1] = (1-\rho)l^i[k] + \gamma J^i[k], \tag{2.3}$$

where ρ is the luciferin decay constant ($0 < \rho < 1$), γ is the luciferin enhancement constant and $J^i[k]$ is the output of the fitness function at the location of glowworm i at timestep k. The second stage is the movement-phase, where each glowworm selects another glowworm based on its 'luciferin' value and moves towards it. A number of different functions can be used for the selection step. Typically, these functions focus on locality. In other words, each glowworm makes selects among its neighbouring glowworms and not the whole swarm. This allows GSO to be used in multi-modal problems, where the swarm can be split into different groups, each one representing a different local maximum of the fitness function. As soon as glowworm i selects to move towards glowworm j, its position $\mathbf{x}^i[k]$ at timestep k is updated using the position update rule

$$\mathbf{x}^i[k+1] = \mathbf{x}^i[k] + s\frac{\mathbf{x}^j[k] - \mathbf{x}^i[k]}{\|\mathbf{x}^j[k] - \mathbf{x}^i[k]\|}, \tag{2.4}$$

where s is the step size, which controls the maximum change in position that can occur for each glowworm. This is one main difference that GSO has from PSO, where the change in position for each particle is unbounded. This difference, along with GSO's ability to detect multiple local maxima can give GSO an advantage over PSO when it comes to the control of robotic swarms for the source localisation.

2.1.5 Firefly Algorithm

The Firefly Algorithm (FA) is another bio-inspired population-based stochastic optimisation algorithm that aims to approximate the behaviour of fireflies in nature. In this algorithm, each firefly emits a virtual glow that represents the fitness of its current location. FA is very similar to GSO, in the sense that the fireflies in the swarm are attracted to brighter fireflies and that it employs locality in order to be able to detect multiple local minima. The difference is that while in GSO, the locality is enforced by the selection process which forces a glowworm to move towards one of its neighbours, in FA, a firefly moves towards all other fireflies. That said, the virtual glow of each firefly diminishes with distance. Therefore, if all fireflies have the same fitness (i.e. same brightness), a firefly will be more attracted towards fireflies that are closer to it, rather

than fireflies that are far away, thereby achieving locality. The rate that the brightness β of a firefly diminishes with distance r is given by

$$\beta = \beta_0 e^{-\gamma r^2}, \tag{2.5}$$

where β_0 is the brightness at distance $r = 0$ and γ is called the light absorption coefficient. Therefore, the change in position $\mathbf{x}^i[k]$ of firefly i at timestep k, due to the attraction towards firefly j is given by

$$\mathbf{x}^i[k+1] = \mathbf{x}^i[k] + \beta_0 e^{-\gamma r_{ij}^2}(\mathbf{x}_j[k] - \mathbf{x}_i[k]) + \alpha \boldsymbol{\epsilon}^i[k], \tag{2.6}$$

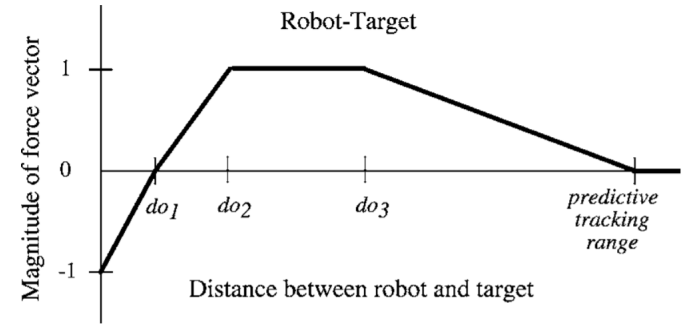
where r_{ij} is the distance between the fireflies i and j. The vector $\boldsymbol{\epsilon}$ is a random vector while the parameter α is a tunable parameter that controls the magnitude of $\boldsymbol{\epsilon}$. The whole term $\alpha\boldsymbol{\epsilon}^i[k]$ is therefore a stochastic component added to the position update equation, to promote exploration. This is another main difference between GSO and FA. In GSO, all stochastic effects exist in the selection process of the neighbouring glowworms. As soon as a neighbouring glowworm is selected though, the position update rule becomes highly deterministic. The stochastic component in FA is an important part of a swarm intelligence algorithm, because it can get the individual unstuck, from situations where a more sophisticated deterministic path planning algorithm would be required. Such stochastic components can be also seen in the position update equation of PSO.

2.1.6 Potential Fields

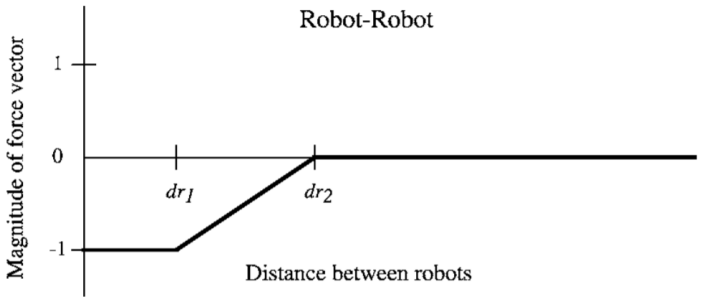
The term potential fields refers to a family of non-bio-inspired techniques that are gaining momentum in the search for control algorithms for swarm robotic applications. In general, these algorithms treat sources of information in the environment (i.e. targets, obstacles, other robots in the swarm etc.) as objects that exert forces (either attractive or repulsive) on each robot of the swarm, guiding its movements. The magnitude of a force is usually dependent on the distance between the robot and the object of interest. One of the first examples of such an algorithm (Parker 2002) aimed to address the problem of Cooperative Multi-Robot Observation of Multiple Moving Targets (CMOMMT). The algorithm is therefore named A-CMOMMT where the letter A stands for ALLIANCE; a formalised approach for the implementation of force vector-fields by the same author (Parker 1994).

The aim of A-CMOMMT is to guide the motion of the swarm, in order to maximise the number of targets that are directly observed by at least one robot in the swarm

at all times. The direction of motion of each robot is calculated using a combination of two different types of forces. Attractive forces are used to guide the robots towards targets, while repulsive forces are used to guide the robots away from other robots of the swarm. Figure 2-3a shows how an attractive force $\mathbf{f}_{i,t}$, exerted by target t on robot i is calculated. Similarly, Figure 2-3b shows how a repulsive force $\mathbf{g}_{i,j}$ exerted by robot j on robot i is calculated.



(a)



(b)

Figure 2-3: Functions defining the magnitudes of virtual force vectors exerted on the robots of the swarm. In (a), the forces are exerted by the targets and they are primarily attractive forces. The region between distances do_2 and do_3 describes the ideal range that the robot needs to maintain from the target. The distance described as "predictive tracking range" signifies the limit beyond which, the target will not attract the robot. At do_1, the attractive force (positive force) becomes repulsive (negative force), to prevent collision of the robot with the target. In (b), the virtual forces are exerted by other robots in the swarm. These forces are always repulsive. Up until distance dr_1, such a force has constant repulsive effect. Beyond this distance, the magnitude of the force drops linearly and becomes 0 at distance dr_2, signifying the maximum distance at which robots can exert forces on other robots.

The direction of motion of robot i is therefore given by

$$\sum_{t=1}^{T} w_{i,t}\mathbf{f}_{i,t} + \sum_{j=1}^{J} \mathbf{g}_{i,j}, \tag{2.7}$$

where T is the number of targets around robot i and J is the number of other robots around robot i. The parameter $w_{i,t}$ represents a weight, used to signify the intention of robot i to move towards target t. In contrast to previous SIA that employed parameters which were only tuned once, prior to the operation of the algorithm, the weights in A-CMOMMT are meant to be dynamically altered during the operation of the swarm.

2.1.7 SIA Applications to Swarm Robotics

So far this chapter has presented the most relevant SIAs for the tasks of source localisation and area searching. The rest of this section will present different swarm robotic variations of these SIA and how they have been applied to different source localisation and area searching scenarios.

Modelling the Individual Behaviours

SIAs can be used for the control of robotic swarms, by associating the motion of simulated agents (e.g. particles, ants, bees etc) of a simulated swarm, with the motion of individual robots in a physical swarm. That said, since many SIA were originally, designed for use in numerical optimisation tasks, they may not be readily applicable to real-world robotic applications in their original form. Therefore, several variations have been proposed that have been applied to different robotic scenarios. An overview of ways that SIAs have been modified and used to directly control the motion of robotic swarms will now be introduced.

dPSO: Hereford et al. (2007) tried to control a physical swarm of up to three robots (mitEBots), using a modified version of PSO, called distributed-PSO (dPSO), with the aim to detect the brightest spot of light in a room. The robots were equipped with light sensors and the fitness of each location was selected based on the light intensity observed at that location.

Hereford et al. (2007) recognise an important problem of the original PSO algorithm that prevents it from directly being used to control physical swarms. In the original algorithm, the motion of particles is assumed to be unconstrained (e.g. the velocities of particles are unbounded both in terms of magnitude and direction). The physical robots used on the other hand, have limited mobility (e.g. a limited turning radius, a

small maximum velocity and cannot move backwards) and therefore they may not be able to react in the way requested by the PSO algorithm. To address this, dPSO does not directly control the velocity of the robots. Instead, the output of dPSO is used to determine a desired location that the robot tries to move towards in the next timestep and the robot moves as close to it as it is permitted by its physical limitations.

A modification is also employed to limit the amount of communication needed by the robots. In original PSO, each particle shares its personal best location with the rest of the swarm at every timestep. This can result in a demand for a large exchange of information that the swarm may not be able to support. To address this, a robot will refrain from sharing its personal best location, if it is the same as in the previous timestep. In other words, a specific personal best location is only shared the first time it is selected by the robot.

Additionally, to further limit the amount of information exchanged, a soon as a new personal best location is identified, only its fitness is shared with other robots (i.e. the actual location is not shared yet). The other robots then examine whether this new personal best location fitness is higher than the global best location fitness that they currently know about and if it is, they request for the actual location information to be transmitted.

Finally, the original PSO does not describe collision avoidance or how the robots should behave if a collision happens. To address this, a behaviour was hardwired when a collision occurred - the robots were programmed to back up and turn rightwards.

The experiments were performed using parameter values of $c_1 = 2$, $c_2 = 2$ and $\omega = 1$. Different experiments were run using different number of robots (1 to 3 robots) and both with and without obstacles. Each experimental case was repeated 10 times and in all cases, the swarms had from 60% to 100% success rate in finding the brightest spot. The time taken to find the brightest spot varied significantly for different repetitions of the same case. Figure 2-4 shows the results of one of the experiments carried out using a single robot. In the figure, another difference of dPSO with the original PSO can be seen. Since the robot is not able to move straight towards its personal best location, due to its limited limited turning radius, it needs to execute a circular manoeuvre to align itself properly with it. During this process, the robot explores the search space further allowing it to discover new locations on its own. In this way, a single robot is capable of exploring the search space and localising the global maximum of the fitness function on its own.

In contrast, in the original algorithm, a particle moves straight to the personal/global

best location and as soon as it reaches it, has no reason to move away, needing to wait for other particles to explore the search space and provide it with a new global best location. Therefore, the original PSO is incapable of proper exploration when the number of particles is very small. This is a useful capability of dPSO that could be utilised in the future to enhance exploration. Nevertheless, such capability should be realised in a predictable and controllable manner and not as a result of the robot not being able to directly follow the commands of the PSO controller (as it is the case with dPSO).

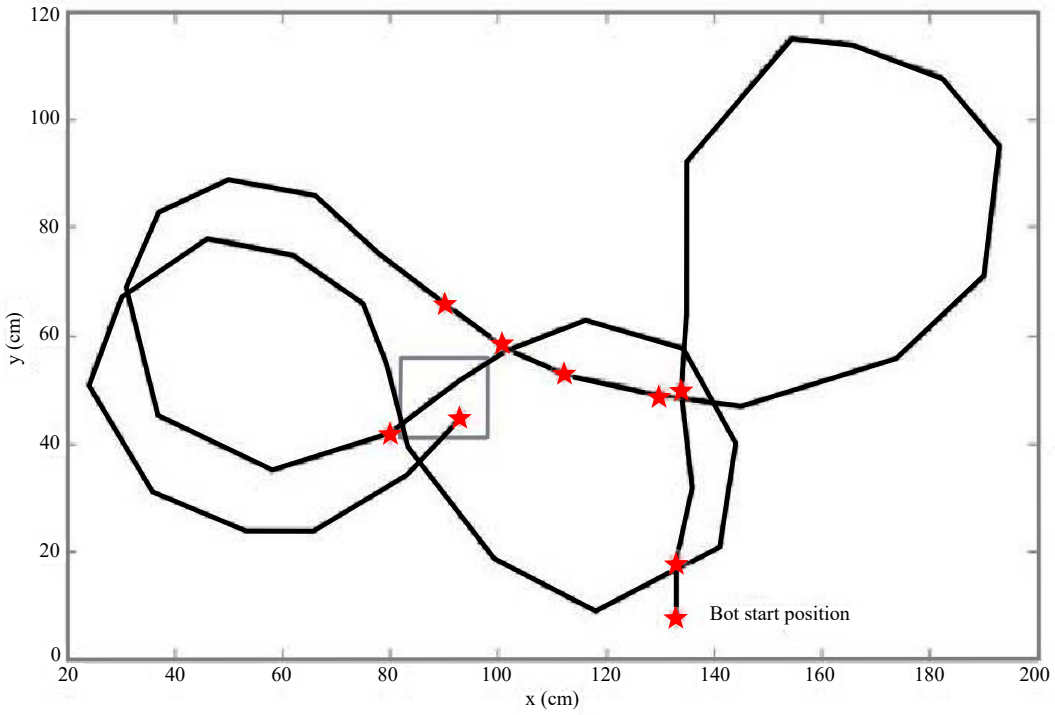


Figure 2-4: Results of the experiments carried out to assess the performance of the dPSO algorithm, using 1 robot, without obstacles. The black lines represents the route followed by the robot and the red square is the global maximum of the fitness function (brightest spot). The red points represent consecutive desired locations towards which the robot needs to move and which are calculated using dPSO. The circular shape of the robot's route is the result of its motion limitations (i.e. limited turning radius) (Hereford et al. 2007).

Apart from the described experiments, other variations of dPSO were further used for the control of robotic swarms in simulated environments. One variant (Perreault et al. 2014) shows how dPSO could be adapted for use in scenarios where communication

with all robots of the swarm is not always possible, while a second variant (Du 2020) extends this approach to also achieve efficient communication with a stationary base and multi-source tracking.

Modified-PSO: Pugh & Martinoli (2007) created a modified-PSO algorithm for use in multi-robot stationary source searching, which was tested in 2D simulations, using the robotic simulator Webots (Michel 2004). The fitness of each location is calculated based on the intensity of the received signal. Like Hereford et al. (2007), they also recognised that PSO assumes unconstrained motion of particles. However they chose to ignore this and assume that the robots can move towards a new direction immediately.

They also identified an additional ambiguity in original PSO that they call *Discrete versus Continuous Time*. PSO particles move in discrete timesteps, while real-world robots move in continuous motion and a PSO timestep can correspond to different time intervals in the real-world. The behaviour of the swarm can therefore change depending on the real-world time interval associated with a PSO timestep.

As in (Hereford et al. 2007), the modified-PSO algorithm, acts more like a high-level trajectory planning search algorithm rather than a low-level motion controller. The algorithm provides the robot with a velocity and the robot is given a pre-set time duration (corresponding to a single timestep) to move towards that direction. At the end of each timestep, the robots check if they have collided with something. Also, in a similar manner to (Hereford et al. 2007), the robots react to collisions, using pre-defined behaviours, independent of the PSO algorithm.

The simulated environment used is of size 8×8 m and the robots have diameter 0.0265 m. Finally, the algorithm is tested for different numbers of robots in the swarm (1-20) and different maximum communication range values. The results successfully show that the performance of the swarms (average distance from the target after 100 seconds) improves with a larger number of robots and increased communication range.

RPSO and RDPSO: Robotic-PSO (RPSO) (Couceiro et al. 2011) is a novel algorithm that was created in an attempt to incorporate collision avoidance directly into PSO. The algorithm achieves this through the introduction of an additional accelerating term in the PSO velocity update equation, which in turn takes the following form

$$\mathbf{u}[k+1] = \omega \mathbf{u}[k] + c_1 \mathbf{r_1} \circ (\mathbf{y}[k] - \mathbf{x}[k]) + c_2 \mathbf{r_2} \circ (\mathbf{y_g}[k] - \mathbf{x}[k]) + \underbrace{c_3 \mathbf{r_3} \circ (\mathbf{F_t}[k])}_{\text{new term}}. \quad (2.8)$$

Here, c_3 is called the obstacle susceptibility coefficient and each element of $\mathbf{r_3}$ is drawn from the uniform distribution $U(0,1)$. The vector $\mathbf{F_t}$ is used to point the robot away from obstacles and other robots. The rest of the parameters are identical to the ones used in original PSO.
In contrast to the cognitive coefficient c_1 and social coefficient c_2, the obstacle susceptibility coefficient c_3 is not meant to have a constant value throughout the operation of the algorithm. Instead, its value varies from 0 when the robot is far away from obstacles, to a maximum value as the robot gets closer to an obstacle.

The algorithm was tested in 2D simulations with simplified physics, where the robots are assumed to be capable of unconstrained movement. From the simulations, it was observed that when a robot is close to obstacles (i.e. c_3 is maximised), if $c_3 \ll \max\{c_1, c_2\}$, the first two accelerating terms of (2.8) (i.e. the cognitive and social terms), overshadow the effect of the last term (i.e. the obstacle susceptibility term), resulting in collisions. On the other hand, if $c_3 \gg \max\{c_1, c_2\}$, collisions are successfully avoided but the robots find it hard to pass through openings and progress towards the source.

To address these problems, Couceiro et al. (2011) make use of another PSO variant, called Darwinian-PSO (DPSO) (Tillett et al. 2005), which is used in numerical optimisation. In DPSO, at every timestep, particles with low fitness have a chance of being removed from the simulation, while particles with the high fitness have a chance of spawning another particle in their general area. When RPSO is combined with DPSO, the Robotic-Darwinian-PSO (RDPSO) is formed.

In RDPSO, instead of being deleted, robots with low fitness are being "socially excluded". In this state, the robots do not try to converge to the source and instead roam randomly in the environment. At the same time, socially excluded robots with high fitness can be again "socially included", at which point they stop roaming randomly and start converging towards the source. In this way, social exclusion/inclusion becomes an exploration mechanism that can also serve to get the robots unstuck from local minima.

The use of RDPSO on robotic swarms has been demonstrated in real-world experiments using a swarm of 12 small robots (eSwarBots) (Couceiro et al. 2013) and simulations of a marine robotic environment exploration scenario where the algorithm is used to control a swarm of autonomous mobile nodes in a mobile ad hoc network (Couceiro et al. 2013, Griffiths Sànchez et al. 2018).

MGSO: In the original GSO algorithm the velocity of glowworms changes using a constant step-size s. This is used to limit the maximum acceleration and velocity that

the algorithm can request from the robots, thereby avoiding the problems of PSO that were previously described. That said, maintaining this constant step-size throughout the operation of the system may not be always desirable. When s is large, the swarm can cover large distances quickly, but after partial convergence to the source has been achieved, it may be impossible to converge fully, since the robots may end up oscillating on top of the source. On the other hand, when s is small, it can take a lot of time for the swarm to converge to the source.

To address these problems, modified-GSO (MGSO) was introduced, that aims to incorporate a variable step-size s into GSO. Zhang et al. (2011) proposed a searching method based on MGSO for plume tracing and odor source localisation. Gupta & Bayal (2020) made use of a similar MGSO algorithm, for the detection of oil spills, using a swarm of marine robots.

Behavioural Parameter Optimisation using Swarm Intelligence Algorithms

The previous section described applications where SIAs were directly controlling the behaviour of each individual robot. Another way to use SIAs is to optimise the parameters of other heuristic algorithms. In other words, since many SIAs were originally designed for numerical optimisation, they can be used in this way to optimise the parameters of other heuristic algorithms, which are in turn used to control the behaviour of the robots. This is a perfectly valid way of using the SIAs but an important disadvantage of such methods needs to be noted. When SIAs are used like this, their inherent scalability, adaptability and robustness characteristics are not inherited by the physical swarm. Furthermore, numerical optimisation in such techniques is not limited to SIAs and other algorithms that are traditionally used in numerical optimisation, such as Genetic Algorithms (GA), can also be used here. This section will describe such approaches for the control of robotic swarms.

Meng & Gan (2008) proposed a collective construction task, where building blocks are randomly distributed in the environment. The robots need to search for the blocks efficiently (each robot searches sub-areas that have not been searched by another robot) and need to cooperate to transport the blocks in predefined locations. Robots share information about discovered blocks by leaving pheromone trails. The robots explore the Grid-based map, identify the location of blocks and assign a utility to each block based on its size. Each robot then selects a block to move, using a modified PSO algorithm that considers the pheromone trail-based utility value of each block. The modified PSO algorithm used (called MS-PSO) was shown to perform better in this dynamically changing environment (when a block has been moved, the information

about its previous position should be ignored, which is represented by the pheromone trail fading out over time).

Pugh & Martinoli (2008) introduced an adaptive strategy for localisation of multiple targets. For this method, each source emits artificial pheromones that fade out as they get further away from the source. The strategy uses several parameters which are optimised using a noise-resistant version of the PSO algorithm that averages the current and previous fitness values to obtain a more robust representation of the actual fitness. The authors ran three different tests to explore how the particle neighbourhood size can affect the performance of the PSO algorithm, by replacing the global best location of the original PSO algorithm with other variables obtained from a predefined number of closest neighbours:

- gbest: all robots are considered as neighbours
- lbest: only the two closest robots are neighbours
- ibest: the 5 closest robots at each side (10 in total) are considered as neighbours

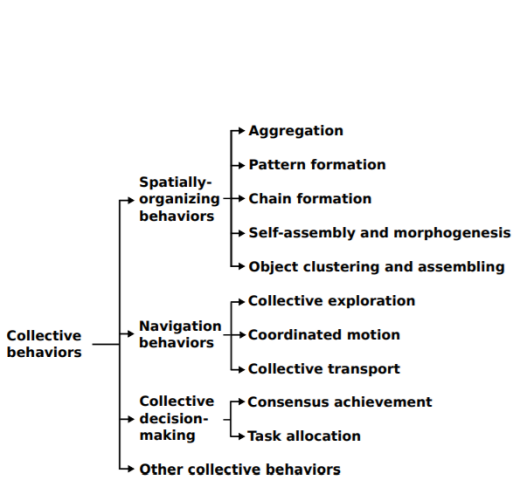
Their results showed that no specific neighbourhood outperforms the rest in the long run.

Oh & Suk (2010) proposed the use of an Artificial Neural Network Controller optimised using a Genetic Algorithm, in order to avoid the difficulties that arise with a logical approach for the implementation of behavioural rules. The network was shown to exceed in performance behaviour-based heuristic controllers. For the basic travelling behaviour of simulated unmanned air vehicles (UAVs), a UAV dynamics model was designed based on a 3 degrees-of-freedom point mass model (forward linear motion, vertical angular motion and horizontal angular motion).

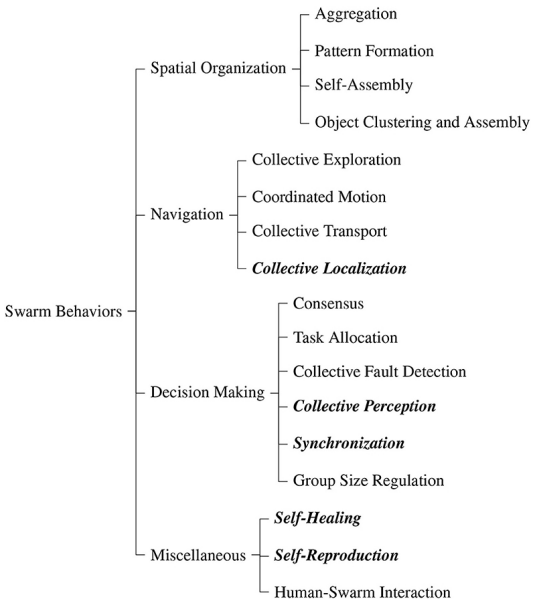
2.2 Swarm Robotic Tasks

The SIAs described in Section 2.1 are algorithms that were designed to be used for the general control of robotic swarms in several different scenarios. SIAs are designed to be operationally simple; a necessary characteristic to maintain operational efficiency and adaptability to several different scenarios, while also avoiding excessive complexity so that scalability and robustness can be ensured. For this reason though, SIAs cannot be expected to provide full control of the robots for all different types of tasks that the swarm may need to address. Therefore, SIAs need to be combined with other heuristic algorithms that are specifically designed to address certain tasks. Numerous

swarm robotic tasks have been identified including aggregation, flocking or coordinated motion, pattern formation, task allocation, collective transport of objects, collective mapping/exploration etc. (Bayindir & Sahin 2007, Bayindir 2015, Nedjah & Junior 2019, Dias et al. 2021). Due to the relatively recent formation of swarm robotics as an engineering discipline, there does not exist a single clear taxonomy of all swarm robotic tasks and collective behaviours. Figure 2-5 shows two different taxonomies that have been proposed in the literature. The taxonomy shown in Figure 2-5b is an adapted version of the one shown in Figure 2-5a. The two taxonomies share many tasks such as aggregation, pattern formation, coordinated motion and task allocation that are classified in the same way. That said, with the advancement of the field, new tasks have also been introduced that were found to be useful in different swarm robotic scenarios, such as self-assembly, collective localisation (i.e. localisation of robots relative to each other) and human-swarm interaction. The rest of this section will describe the tasks that are mostly relevant to scenarios of source localisation and monitoring.



(a) Brambilla et al. (2013)



(b) Schranz et al. (2020)

Figure 2-5: Two different taxonomies used to define swarm robotic tasks/behaviours. The taxonomy presented in (b) is an adapted version of the one presented in (a). Primary tasks such as aggregation, pattern formation, coordinated motion and task allocation appear in both taxonomies, but several new tasks are also introduced in (b) such as self-assembly, collective localisation and human-swarm interaction. These new behaviours signify the research advancements and the appearance of new questions regarding this field in recent years.

2.2.1 Aggregation

One of the most important tasks of a robotic swarm is aggregation, i.e. the ability to form and maintain groups of robots that remain close while avoiding collisions with each other (Bayindir & Sahin 2007, Nedjah & Junior 2019). For example, when two robots are too close to each other, an aggregation algorithms will cause them to drift apart. On the other hand, when a robot is too far away from the rest of the swarm, aggregation forces will pull it back towards the rest of the robots, before it is too far away and out of communication range.

2.2.2 Flocking

Flocking of coordinated motion focuses on the collective movement of the swarm, efficiently, over large distances. The techniques that are used for flocking can be often similar to the ones used for aggregation in the sense that artificial physics may be used to keep the robots close to each other. An important component of flocking that does not exist in aggregation is velocity consensus (i.e. the robots of the swarm collectively decide on and share the same velocity magnitude and direction), while it can also sometimes incorporate some type of pattern formation technique. Flocking is important in source tracking, since it can allow the swarm to cover the search area more efficiently and chase the source easily if it moves away.

Olfati-Saber (2006) initially tried to create a flocking algorithm that follows the three rules required for flocking behaviour to emerge, as stated by Reynolds (1987), using principles of graph theory. The three rules are:

- Flock Centring: Maintaining close proximity to nearby members of the flock.
- Collision Avoidance: Avoiding collision with nearby objects.
- Velocity Consensus: Matching velocity with nearby members of the flock.

After trying to express the three rules in a mathematical form, as part of the control protocol of an algorithm, they showed that they are insufficient for flocking behaviour to emerge. Instead, they propose a new algorithm that includes one additional navigational feedback term to ensure that all agents in a flock have a common target objective.

The control protocol of the proposed algorithm is shown in Equation (2.9), where the first term $\sum_{j \in N_i} \phi_\alpha(||r_j - r_i||_\sigma) e_{ij}$ is a gradient-based portion that applies the first two rules, the second term $\sum_{j \in N_i} a_{ij}(v_j - v_i)$ achieves the velocity consensus of the third rule and the last term $c_1(r_t - r_i) + c_2(v_t - v_i)$ is the navigational feedback term.

$$u_i = \sum_{j \in N_i} \phi_\alpha(||r_j - r_i||_\sigma) e_{ij} + \sum_{j \in N_i} a_{ij}(v_j - v_i) + c_1(r_t - r_i) + c_2(v_t - v_i), \qquad (2.9)$$

where u_i is the control input, N_i is the set of neighbouring agents, r_i is the position vector and v_i is the velocity vector of agent i, r_t and v_t are the position vector and velocity vector of the target and ϕ_α is the action-force function shown in Figure 2-6, where r_α and d_α are constant parameter representing the ideal separation between adjacent agents and the maximum communication range of an agent respectively. Additionally, a_{ij} is a binary variable that has value 1 when the agents i and j are part of the same flock and 0 otherwise. $c_1 > 0$ and $c_2 > 0$ are constants.

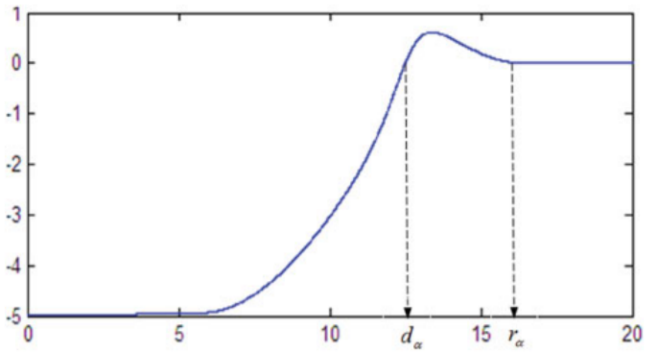


Figure 2-6: The ψ_α action force function is used to calculate the forces between agents of the swarm, based on the distance z between pairs of agents, in order to maintain the desired spatial separation (Olfati-Saber 2006).

The proposed algorithm was tested in both 2-D and 3-D simulations, for target tracking and obstacle avoidance scenarios. It was shown to be capable of quick self-organising of the flock, smooth de-aggregation and re-aggregation during obstacle avoidance, squeezing to pass through narrow gaps, all while avoiding fragmentation into smaller groups when this is not desirable. Figure 2-7 shows six instances of an obstacle avoidance simulation where the swarm is required to split in order to avoid obstacles and rejoin afterwards.

Huiqin et al. (2015) recognised that the algorithm proposed by Olfati-Saber (2006) does not allow flocking while tracking multiple targets, because the whole swarm cannot be separated into smaller groups. To address this problem they incorporated anti-flocking behaviour (Miao et al. 2010) to the algorithm as a means to encourage exploration of other targets. In more detail, each agent can be in one of the two states 1) flocking, where the agent will aim to become part of a flock that follows a target by being attracted to its closest targets, as in (Olfati-Saber 2006) and 2) anti-flocking, where

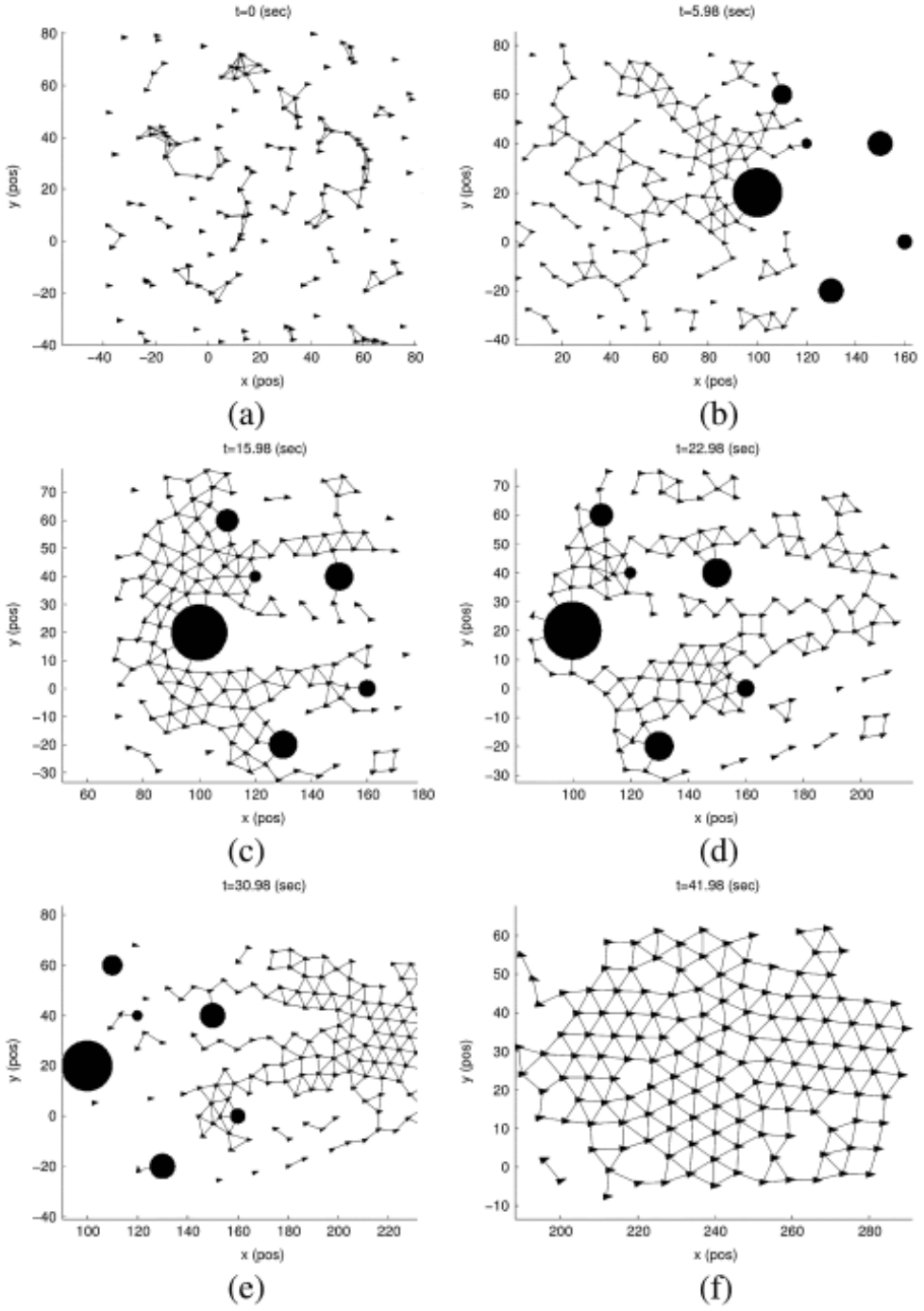


Figure 2-7: Instances of an obstacle avoidance simulation. In (a), the swarm is initialised and the agents have not yet formed a solid group. In (b) and (c), the agents begin to come together, maintaining constant distance from neighbours and velocity consensus. In the presence of obstacles the agents separate to avoid collision. In (d) to (f), the agents rejoin together into a solid group after the obstacles have been successfully avoided. (Olfati-Saber 2006).

if an agent is not in a close distance to a target, it is greedily attracted towards its closest target, without trying to maintain flock centring and velocity consensus with other robots. Aggregation dynamics are also applied to this latter state in order to avoid collision with other explorers.

2.2.3 Target Entrapment

The escort/entrapment problem, also known as catch problem, is a task of multi-robot systems that revolves around entrapping a moving target by reducing the separation between robots, in order to prevent it from escaping or protect it from intruders (Antonelli et al. 2007). This is a crucial task after localisation of the source has been achieved, since it can allow the swarm to encircle and monitor it from a safe distance. The number of papers that aimed to solve this problem in the past are limited, and most of the earlier literature addressed it by aiming to achieve group formations based on ordinary shapes like circles, ellipses and rectangles (Antonelli et al. 2007)(Barnes et al. 2009)(Escobedo et al. 2014). This results in major disadvantages because it limits the pattern-formation flexibility of the system, especially during obstacle avoidance and multi-target entrapment.

Zhang et al. (2018) proposed a novel heuristic solution to the target entrapment problem. Their method allows a swarm of robots to trap single targets or aggregated targets effectively by generating an implicit function that aims to describe an entrapment formation pattern of irregular shape. The shape should not contain unnecessary empty space, while including all targets. It should also be adjustable to exclude possible obstacles. Figure 2-8 shows an example of such an entrapment formation. To achieve this, the algorithm creates three sets of points (internal, boundary and external points). The implicit pattern function $f(g)$ is calculated using a modification of the Gene Regulatory Network (GRN) controller, which is borrowed from the field of life sciences and it is used to describe mathematically the relationship of genes to proteins, during the process of morphogenesis, i.e. the process of growing natural tissue and organs into specific shapes.

The algorithm was tested on both static and moving targets (including de-aggregation and re-aggregation of targets), as well as on obstacle avoidance scenarios, with seemingly good results. The swarm is capable of splitting into different groups during the de-aggregation of the trapped targets and merge back to a single group when the targets re-aggregate. It is also capable of "squeezing" appropriately to pass through narrow openings, while maintaining the entrapment shape. That being said, the authors point to room for improvement with regards to the uncontrollable number of robots required

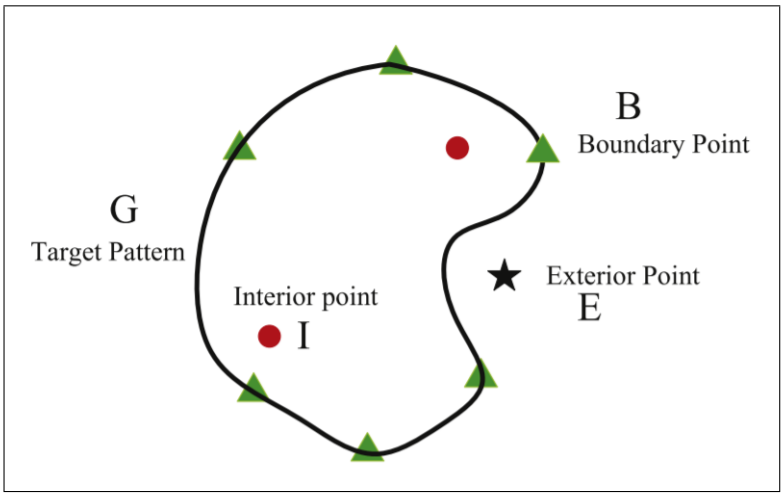


Figure 2-8: Implicit Formation Pattern Function that surrounds all targets (Zhang et al. 2018)

to form the formation pattern, which can result to more robots than necessary being used.

2.2.4 Multi-Target Tracking

Multi-target tracking is an important task in source localisation, as it allows a source localisation system to identify multiple sources. In real-world scenarios it can be rarely guaranteed that only a single source will exist inside the search area. Most of the SIAs presented in Section 2.1 have variants that can allow them to perform multi-modal searching (i.e. multi-source localisation). Alternatively, a number of other interesting techniques exist outside the field of swarm robotics, which can be used to provide more information about the individual sources. However, it should be pointed out that the following techniques were not designed to be inherently scalable, adaptable and robust and therefore their adaptation to robotic swarms may not be readily achievable.

Multiple-Input-Multiple-Output (MIMO) systems are multi-target detection and classification systems that recently have been receiving increasing attention from the marine engineering community for their ability to detect and track multiple targets in a cluttered environment by differentiating them from false positives as well as being able to classify them.

In order to correctly carry out the data association tasks required for the correct operation of these systems, several sophisticated algorithms have been proposed. One of the most successful was the Multiple Hypothesis Tracking (MHT) (Blackman 2004)

algorithm, an algorithm tracker that performs data association probabilistically. The disadvantage of MHT and in general most algorithms that build on Kalman Filters is that they rely on heuristics. Another algorithm called the Probability Hypothesis Density (PHD) (Vo et al. 2003, Vo & Ma 2006, Mahler 2007) algorithm, relies on spatial statistic tools such as point processes, without requiring any use of heuristics but with the shortcoming that the algorithm cannot provide target classification.

To overcome the limitation of both algorithms, while maintaining their advantages, a novel algorithm called the Hypothesised filter for Independent Stochastic Populations (HISP) (Houssineau 2015) was introduced. The HISP filter follows the ideas of operation of the PHD filter, but target detection and localisation are performed while distinguishing the targets. This allows target classification to be added to the algorithm naturally, without further required processing. HISP is a relatively recent filter so further research needs to be done on its performance but its operational capabilities have already been studied for different applications (McKenna et al. 2015) (Pailhas et al. 2017).

2.3 Discussion of Current Literature

So far, this chapter has presented the core concepts of swarm robotics and swarm intelligence that can be used to design and implement a robotic swarm for the task of source localisation and monitoring. In general, it is expected that a fully functioning robotic swarm would need to combine several different methods, namely a SIA for the localisation of a source, as well as techniques such as aggregation and collision avoidance to keep the robots close to each other while avoiding collisions, flocking to allow the robots to travel efficiently over long distances and some type of source entrapment technique to encircle the source and allow its monitoring from a safe distance. Ideally, the system should also be capable of multi-source tracking, either through the use of an appropriately modified SIA or any other technique from the ones discussed in Section 2.2.4.

The literature offers several examples where 2 or 3 techniques may be combined together (typically aggregation or flocking are combined with another technique) (Olfati-Saber 2006, Olfati-Saber & Jalalkamali 2011). That said, there exist no studies, where robotic swarms (either simulated or physical) are shown to be able to combine 4 or more tasks as would be ideally required by the scenario studied in this thesis. A reason behind this is given by Bayindir (2015):

> The possibility to achieve global objectives at the swarm level by means of dis-

tributed algorithms acting at the individual level comes at a price: it is often difficult to design the individual robot behaviour so that the global performance is maximized.

In other words, the ability to achieve emergent collective global behaviour out of individual local decisions adds complexity into the methodologies used in swarm robotics. When different techniques are merged together, this complexity can increase significantly and extra effort is required to ensure that all techniques remain operational as well as scalable, adaptable and robust. An example of this problem can be seen in RPSO (Couceiro et al. 2011) (see Section 2.1.7), where it is described how a perfect balance is required between the RPSO parameters to ensure that both the source localisation and obstacle avoidance properties of RPSO can be utilised properly, without interfering with each other.

Another similar problem can be also seen from the use of SIA. In Section 2.1.7, all of the reported studies mention that SIA do not consider the movement limitations that may restrict the motion of the physical robots. As a result, SIA are typically used as high-level trajectory planning algorithms, rather than actual low-level motion controllers. This can further interfere with the operation and scalability of the SIA.

Based on the aforementioned problems, it can be concluded that a low-level motion controller for robotic swarms is required, which will allow both the easy merging of different techniques while also taking into consideration the physical limitations of the robots. Such a controller could eventually lead to the standardisation of swarm robotic techniques, a milestone that has not been achieved yet by the current literature.

Another problem that prevents techniques from being merged together can be seen when comparing source localisation and target entrapment methods. In theory, these two types of methods should be designed to operate together, in the sense that the swarm makes use of a SIA for collaborative source localisation and as the swarm gets close to the source, a target entrapment technique such as the ones presented in Section 2.2.3 can be employed so that the robots maintain a secure distance from the source while monitoring it. The problem, as stated by Pugh & Martinoli (2007), is that, in order for a SIA to know the location of the source, a robot must pass over it. On the other hand, all of the target entrapment techniques provided, either assume prior knowledge of the source location, or that the robots can identify the source location on their own in a simple way. Therefore, current SIAs and target entrapment techniques cannot be combined together at this point. To solve this problem, a SIA needs to be modified in a way that will allow it to identify the location of the source, without the need for a

robot to pass on top of it (i.e. localise the source from far away).

2.4 Conclusion

This chapter has presented swarm robotic control algorithms and methodologies that are relevant to the scenario of source localisation and monitoring. The types of methodologies presented are swarm intelligence algorithms, that can be primarily used for the task of source localisation, aggregation and flocking techniques which are mainly concerned with keeping the robots of the swarm close to each other and target entrapment techniques that are used to encircle a localised source and maintain a specific distance from it. Multi-target tracking techniques were also presented. Finally, several problems and gaps that exist in the literature were identified and discussed. Namely, there is a lack of a SIA capable of accurate motion control of the robots by considering their movement limitations, a way to readily merge a large number of swarm robotic tasks for applications that require it, and a SIA capable of localising a source from far away to allow its merging with current source entrapment techniques. It is believed that solving these problems will open the way for the design of a fully functioning robotic swarm capable of underwater source localisation and monitoring, at least from the robotic swarm control perspective. This thesis will focus on addressing these problems through the modification of a SIA.

References

Abraham, A. & Ramos, V. (2003), Web usage mining using artificial ant colony clustering and linear genetic programming, *in* 'The 2003 Congress on Evolutionary Computation, 2003. CEC '03.', Vol. 2, pp. 1384–1391 Vol.2.

Antonelli, G., Arrichiello, F. & Chiaverini, S. (2007), The entrapment/escorting mission for a multi-robot system: Theory and experiments, *in* '2007 IEEE/ASME international conference on advanced intelligent mechatronics', pp. 1–6.

Barnes, L. E., Fields, M. A. & Valavanis, K. P. (2009), 'Swarm Formation Control Utilizing Elliptical Surfaces and Limiting Functions', *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* **39**(6), 1434–1445.

Bayindir, L. (2015), 'A Review of Swarm Robotics Tasks', *Neurocomputing* **172**.

Bayindir, L. & Sahin, E. (2007), 'A review of studies in swarm robotics', *Turkish Journal of Electrical Engineering and Computer Sciences* **15**, 115–147.

Beni, G. (2005), From Swarm Intelligence to Swarm Robotics, *in* 'Swarm Robotics', Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 1–9.

Beni, G. & Wang, J. (1993), Swarm Intelligence in Cellular Robotic Systems, *in* P. Dario, G. Sandini & P. Aebischer, eds, 'Robots and Biological Systems: Towards a New Bionics?', Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 703–712.

Biesmeijer, J. C. & Seeley, T. D. (2005), 'The use of waggle dance information by honey bees throughout their foraging careers', *Behavioral Ecology and Sociobiology* **59**(1), 133–142.
URL: *https://doi.org/10.1007/s00265-005-0019-6*

Blackman, S. S. (2004), 'Multiple hypothesis tracking for multiple target tracking', *IEEE Aerospace and Electronic Systems Magazine* **19**(1), 5–18.

Blackwell, T. (2007), Particle Swarm Optimization in Dynamic Environments, *in* S. Yang, Y.-S. Ong & Y. Jin, eds, 'Evolutionary Computation in Dynamic and Uncertain Environments', Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 29–49.

Bonyadi, M. R. & Michalewicz, Z. (2016), 'Stability Analysis of the Particle Swarm Optimization Without Stagnation Assumption', *IEEE Transactions on Evolutionary Computation* **20**(5), 814–819.

Brambilla, M., Ferrante, E., Birattari, M. & Dorigo, M. (2013), 'Swarm Robotics: A Review from the Swarm Engineering Perspective', *Swarm Intelligence* **7**, 1–41.

Bullnheimer, B., Hartl, R. F. & Strauss, C. (1999), Applying the ANT System to the Vehicle Routing Problem, *in* S. Voß, S. Martello, I. H. Osman & C. Roucairol, eds, 'Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization', Springer US, Boston, MA, pp. 285–296.

Carlisle, A. & Dozier, G. (2000), 'Adapting Particle Swarm Optimization to Dynamic Environments', *Proc of Int Conf on Artificial Intelligence* .

Chakraborty, A. & Kar, A. K. (2017), Swarm Intelligence: A Review of Algorithms, *in* S. Patnaik, X.-S. Yang & K. Nakamatsu, eds, 'Nature-Inspired Computing and Optimization: Theory and Applications', Springer International Publishing, Cham, pp. 475–494.

Cleghorn, C. W. & Engelbrecht, A. P. (2018), 'Particle swarm stability: a theoretical extension using the non-stagnate distribution assumption', *Swarm Intelligence* **12**(1), 1–22.

Couceiro, M. S., Rocha, R. P. & Ferreira, N. M. F. (2011), A novel multi-robot exploration approach based on Particle Swarm Optimization algorithms, *in* '2011 IEEE International Symposium on Safety, Security, and Rescue Robotics', pp. 327–332.

Couceiro, M. S., Rocha, R. P. & Ferreira, N. M. F. (2013), 'A PSO multi-robot exploration approach over unreliable MANETs', *Advanced Robotics* **27**(16), 1221–1234.

Dias, P. G. F., Silva, M. C., Rocha Filho, G. P., Vargas, P. A., Cota, L. P. & Pessin, G. (2021), 'Swarm Robotics: A Perspective on the Latest Reviewed Concepts and Applications', *Sensors* **21**(6).

Dorigo, M., Birattari, M. & Stutzle, T. (2006), 'Ant colony optimization', *IEEE Computational Intelligence Magazine* **1**(4), 28–39.

Du, Y. (2020), 'A Novel Approach for Swarm Robotic Target Searches Based on the DPSO Algorithm', *IEEE Access* **8**, 226484–226505.

Eberhart & Shi, Y. (2001), Particle swarm optimization: developments, applications and resources, *in* 'Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)', Vol. 1, pp. 81–86 vol. 1.

Escobedo, R., Muro, C., Spector, L. & Coppinger, R. P. (2014), 'Group size, individ-

ual role differentiation and effectiveness of cooperation in a homogeneous group of hunters', *Journal of The Royal Society Interface* **11**(95).

Fernandez-Marquez, J. L. & Arcos, J. L. (2009), An Evaporation Mechanism for Dynamic and Noisy Multimodal Optimization, *in* 'Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation', GECCO '09, Association for Computing Machinery, New York, NY, USA, pp. 17–24.

Gambardella, L. M. & Dorigo, M. (1996), Solving symmetric and asymmetric TSPs by ant colonies, *in* 'Proceedings of IEEE International Conference on Evolutionary Computation', pp. 622–627.

Grassé, P.-P. (1959), 'La reconstruction du nid et les coordinations interindividuelles chezBellicositermes natalensis etCubitermes sp. la théorie de la stigmergie: Essai d'interprétation du comportement des termites constructeurs', *Insectes Sociaux* **6**(1), 41–80.

Griffiths Sànchez, N. D., Vargas, P. A. & Couceiro, M. S. (2018), A Darwinian Swarm Robotics Strategy Applied to Underwater Exploration, *in* '2018 IEEE Congress on Evolutionary Computation (CEC)', pp. 1–6.

Gupta, R. & Bayal, R. K. (2020), Source Detection of Oil Spill using Modified Glowworm Swarm optimization, *in* '2020 5th International Conference on Computing, Communication and Security (ICCCS)', pp. 1–6.

Hereford, J. M., Siebold, M. & Nichols, S. (2007), Using the Particle Swarm Optimization Algorithm for Robotic Search Applications, *in* '2007 IEEE Swarm Intelligence Symposium', pp. 53–59.

Houssineau, J. (2015), Representation and estimation of stochastic populations, PhD thesis, Heriot-Watt University.

Huiqin, P., Shiming, C. & Qiang, L. (2015), 'A local flocking algorithm of multi-agent dynamic systems', *International Journal of Control* p. 1.
URL: *http://dx.doi.org/10.1080/00207179.2015.1039595*

Juneja, M. & Nagar, S. K. (2016), Particle swarm optimization algorithm and its parameters: A review, *in* '2016 International Conference on Control, Computing, Communication and Materials (ICCCCM)', pp. 1–5.

Karaboga, D. (2005), 'An Idea Based on Honey Bee Swarm for Numerical Optimization, Technical Report - TR06', *Technical Report, Erciyes University* .

Kazadi, S. (2000), Swarm engineering, PhD thesis, California Institute of Technology.

Kennedy, J. & Eberhart, R. (1995), Particle swarm optimization, *in* 'Proceedings of ICNN'95 - International Conference on Neural Networks', Vol. 4, pp. 1942–1948 vol.4.

Khatib, O. (1985), Real-time obstacle avoidance for manipulators and mobile robots, *in* 'Proceedings. 1985 IEEE International Conference on Robotics and Automation', Vol. 2, pp. 500–505.

Krishnanand, K. N. & Ghose, D. (2009), 'Glowworm swarm optimization for simultaneous capture of multiple local optima of multimodal functions', *Swarm Intelligence* **3**(2), 87–124.

Liu, Q. (2014), 'Order-2 Stability Analysis of Particle Swarm Optimization', *Evolutionary computation* **23**.

Mahler, R. (2007), 'PHD filters of higher order in target number', *IEEE Transactions on Aerospace and Electronic Systems* **43**(4), 1523–1543.

McKenna, I., Tonolini, F., Tobin, R., Houssineau, J., Bridle, H., McDougall, C., Schlangen, I., McGrath, J. S., Jimenez, M. & Clark, D. E. (2015), Observing the Dynamics of Waterborne Pathogens for Assessing the Level of Contamination, *in* '2015 Sensor Signal Processing for Defence (SSPD)', pp. 1–5.

Meng, Y. & Gan, J. (2008), A distributed swarm intelligence based algorithm for a cooperative multi-robot construction task, *in* '2008 IEEE Swarm Intelligence Symposium', pp. 1–6.

Miao, Y., Khamis, A. & Kamel, M. S. (2010), Applying anti-flocking model in mobile surveillance systems, *in* '2010 International Conference on Autonomous and Intelligent Systems, AIS 2010', pp. 1–6.

Michel, O. (2004), 'WebotsTM: Professional Mobile Robot Simulation', *International Journal of Advanced Robotic Systems* **1**.

Nayak, B., Dash, S. K. & Sahu, J. B. (2019), Validation of Well-Known Population-Based Stochastic Optimization Algorithms Using Benchmark Functions, *in* J. C. Bansal, K. N. Das, A. Nagar, K. Deep & A. K. Ojha, eds, 'Soft Computing for Problem Solving', Springer Singapore, Singapore, pp. 731–744.

Nedjah, N. & Junior, L. S. (2019), 'Review of methodologies and tasks in swarm robotics towards standardization', *Swarm and Evolutionary Computation* **50**, 100565.

Oh, S. H. & Suk, J. (2010), Evolutionary design of the controller for the search of area with obstacles using multiple UAVs, *in* 'ICCAS 2010', pp. 2541–2546.

Olfati-Saber, R. (2006), 'Flocking for multi-agent dynamic systems: algorithms and theory', *IEEE Transactions on Automatic Control* **51**(3), 401–420.

Olfati-Saber, R. & Jalalkamali, P. (2011), Collaborative target tracking using distributed Kalman filtering on mobile sensor networks, pp. 1100–1105.

Ozcan, E. & Mohan, C. K. (1999), Particle swarm optimization: surfing the waves, *in* 'Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)', Vol. 3, pp. 1939–1944 Vol. 3.

Pailhas, Y., Houssineau, J., Petillot, Y. R. & Clark, D. E. (2017), 'Tracking with MIMO sonar systems: applications to harbour surveillance', *IET Radar, Sonar Navigation* **11**(4), 629–639.

Parker, L. E. (1994), ALLIANCE: an architecture for fault tolerant, cooperative control of heterogeneous mobile robots, *in* 'Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'94)', Vol. 2, pp. 776–783 vol.2.

Parker, L. E. (2002), 'Distributed Algorithms for Multi-Robot Observation of Multiple Moving Targets', *Autonomous Robots* **12**(3), 231–255.

Perreault, L., Wittie, M. P. & Sheppard, J. (2014), Communication-aware distributed PSO for dynamic robotic search, *in* '2014 IEEE Symposium on Swarm Intelligence', pp. 1–8.

Portugal, D. & Rocha, R. P. (2016), 'Cooperative multi-robot patrol with Bayesian learning', *Autonomous Robots* **40**(5), 929–953.

Pugh, J. & Martinoli, A. (2007), Inspiring and Modeling Multi-Robot Search with Particle Swarm Optimization, *in* '2007 IEEE Swarm Intelligence Symposium', pp. 332–339.

Pugh, J. & Martinoli, A. (2008), Distributed Adaptation in Multi-robot Search Using Particle Swarm Optimization, *in* M. Asada, J. C. T. Hallam, J.-A. Meyer & J. Tani, eds, 'From Animals to Animats 10', Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 393–402.

Reynolds, C. W. (1987), 'Flocks, Herds and Schools: A Distributed Behavioral Model', *SIGGRAPH Comput. Graph.* **21**(4), 25–34.
URL: *http://doi.acm.org/10.1145/37402.37406*

Schranz, M., Umlauft, M., Sende, M. & Elmenreich, W. (2020), 'Swarm Robotic Behaviors and Current Applications', *Frontiers in Robotics and AI* **7**, 36.

Senanayake, M., Senthooran, I., Barca, J. C., Chung, H., Kamruzzaman, J. & Murshed, M. (2016), 'Search and tracking algorithms for swarms of robots: A survey', *Robotics and Autonomous Systems* **75**, 422–434.

Sharkey, A. J. C. (2007), 'Swarm robotics and minimalism', *Connection Science* **19**(3), 245–260.

Spears, W. M., Spears, D. F., Hamann, J. C. & Heil, R. (2004), 'Distributed, Physics-Based Control of Swarms of Vehicles', *Autonomous Robots* **17**(2), 137–162.

Tehrani, K. F., Zhang, Y., Shen, P. & Kner, P. (2017), 'Adaptive optics stochastic optical reconstruction microscopy (AO-STORM) by particle swarm optimization', *Biomed. Opt. Express* **8**(11), 5087–5097.

Theraulaz, G. & Bonabeau, E. (1999), 'A brief history of stigmergy.', *Artificial life* **5**(2), 97–116.

Tillett, J., Rao, T., Sahin, F. & Rao, R. (2005), Darwinian Particle Swarm Optimization., pp. 1474–1487.

Trelea, I. C. (2003), 'The particle swarm optimization algorithm: convergence analysis and parameter selection', *Information Processing Letters* **85**(6), 317–325.

Vo, B. . & Ma, W. . (2006), 'The Gaussian Mixture Probability Hypothesis Density Filter', *IEEE Transactions on Signal Processing* **54**(11), 4091–4104.

Vo, B.-N., Singh, S. & Doucet, A. (2003), Sequential monte carlo implementation of the phd filter for multi-target tracking, *in* 'Sixth International Conference of Information Fusion, 2003. Proceedings of the', Vol. 2, pp. 792–799.

Winfield, A. F. T., Harper, C. J. & Nembrini, J. (2005), Towards Dependable Swarms and a New Discipline of Swarm Engineering, *in* 'Swarm Robotics', Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 126–142.

Zhang, J., Chen, J. & Che, L. (2020), Hybrid PSO Algorithm with Adaptive Step Search in Noisy and Noise-free Environments, *in* '2020 IEEE Congress on Evolutionary Computation (CEC)', pp. 1–8.

Zhang, J., Zhu, X., Wang, Y. & Zhou, M. (2019), 'Dual-Environmental Particle Swarm

Optimizer in Noisy and Noise-Free Environments', *IEEE Transactions on Cybernetics* **49**(6), 2011–2021.

Zhang, S., Liu, M., Lei, X., Huang, Y. & Zhang, F. (2018), 'Multi-target trapping with swarm robots based on pattern formation', *Robotics and Autonomous Systems* **106**, 1–13.
URL: *http://www.sciencedirect.com/science/article/pii/S0921889017306024*

Zhang, Y., Ma, X. & Miao, Y. (2011), Localization of multiple odor sources using modified glowworm swarm optimization with collective robots, *in* 'Proceedings of the 30th Chinese Control Conference', pp. 1899–1904.

Şahin, E. (2005), Swarm Robotics: From Sources of Inspiration to Domains of Application BT - Swarm Robotics, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 10–20.

Chapter 3

Adapted Particle Swarm Optimisation

The use of swarm intelligence algorithms (SIAs) as low-level motion controllers for robotic swarms has been so far difficult and instead they have been limited to being used only for high-level trajectory planning. As discussed in Chapter 2, there are many reasons and limitations that prevent SIAs from being used in this way. Namely, the inability to consider the physical limitations of the individual robots (e.g. maximum velocity, maximum acceleration etc.) and the high complexity that comes with merging an SIA with other techniques. The next three chapters of this thesis will aim to modify an SIA, in order to address these limitations. Later, Chapters 7 and 8 will show how an SIA can be further modified to address additional problems, specific to source localisation scenarios, such as the inability of SIAs to work together with target entrapment techniques.

Due to its popularity and the large number of studies that explain its behaviour, PSO (Kennedy & Eberhart 1995) was selected to be the algorithm that will be used throughout this thesis. A close alternative would have been GSO (Krishnanand & Ghose 2009), as several studies exist about how it could be used in swarm robotics. That said, GSO only makes use of the current location of other robots in the swarm (i.e. robots move towards other robots) and therefore it only allows interaction with other robots. On the other hand, PSO considers specific locations in the world (i.e. personal and global best locations), which if associated correctly with the location of the source, could allow the swarm to interact with the source through the use of a target entrapment technique. It should be noted though that some of the SIAs presented in Chapter 2 (including GSO)

share several similarities with PSO and therefore the results of this chapter may be extendable to them as well. Taking GSO as an example, its position update equation is similar to PSO in the sense that the next position is equal to the previous position plus a single additive term. By controlling the magnitude of the additive term it is therefore possible to control the maximum velocity of the robot. Furthermore, more additive terms can also be included, representing other tasks (e.g. obstacle avoidance). In this case, controlling the combined magnitude of all additive terms allows the limitation of the maximum velocity of the robot. These are the basic ideas that will be explored in the next chapters.

This chapter will introduce a novel PSO variant, modified to address the following problems that prevent the original algorithm from directly controlling the motion of robotic swarms:

1. As recognised by Hereford et al. (2007) and Pugh & Martinoli (2007), the particles in PSO are assumed to be physically unconstrained (i.e. unconstrained velocity and acceleration); an assumption that does not hold for physical robots.

2. As recognised by Pugh & Martinoli (2007), the amount of time (in seconds) that corresponds to one PSO iteration (i.e. one timestep), is currently arbitrarily defined. This can result in different swarm behaviours for different timestep sizes, making them difficult to predict and control.

The next section will introduce the relevant PSO theory in detail, before modifications are introduced to address both problems.

3.1 Particle Swarm Optimisation Theory

PSO aims to identify the location inside a multidimensional space that maximises a fitness function by using a swarm of particles. Let $f\colon \mathbb{R}^d \to \mathbb{R}$ be the fitness function that needs to be maximised, where d is the number of dimensions. Let the particle swarm $\Omega[k]$ be a set of N particles located in $\mathbb{R}^d$, at timestep k. Each particle computes the cost of its current location using f. The movement of the particles is then determined by the position update equation (2.2) and the velocity update equation (2.1) where $\mathbf{u}^i[k]$ and $\mathbf{x}^i[k]$ are the velocity and position of each particle i at timestep k, respectively. The $\circ$ operator represents element-wise multiplication (the Schur product). Each element of the vectors $\mathbf{r_1}$ and $\mathbf{r_2}$ is drawn from the uniform distribution,

$$r_{1,j},\ r_{2,j} \sim U(0,1) \quad 1 \leq j \leq d. \tag{3.1}$$

The location $\mathbf{y}^i$ is the personal best location for particle i, such that, for any timestep $l \in \mathbb{Z}^+,\ l \leq k$

$$f(\mathbf{y}^i[k]) \leq f(\mathbf{x}^i[l]). \tag{3.2}$$

Similarly, $\mathbf{y_g}$ is the global best location, such that, for any particle $i \in [1 : N]$

$$f(\mathbf{y_g}[k]) \leq f(\mathbf{y}^i[k]). \tag{3.3}$$

The parameters ω, c_1 and c_2 are the inertia weight, the cognitive coefficient and the social coefficient respectively (Cleghorn & Engelbrecht 2018). Alternatively, c_1 and c_2 are known as the accelerating coefficients and their corresponding terms in Equation (2.1) are known as the accelerating terms.

The inertia weight ω prevents the particles of the swarm from diverging uncontrollably from either their personal or the global best location. The cognitive coefficient c_1 and social coefficient c_2 control the rate of convergence towards the personal best ($\mathbf{y}$) or the global best ($\mathbf{y_g}$) location. Improper tuning of the PSO parameters can affect the stability of the algorithm (i.e. the particles may diverge uncontrollably) and there have been numerous studies on the parameter values that can guarantee PSO stability. These will be discussed in the following sub-section(Clerc & Kennedy 2002, Kennedy 2010, Shi & Eberhart 1998).

3.1.1 Parameter Tuning

As discussed in Chapter 2, to date it has not been possible to prove mathematically that PSO will eventually converge to the global extremum of its fitness function. That said, several stability analyses have shown that PSO will eventually stochastically converge inside an arbitrary region (i.e. will not diverge uncontrollably) when certain PSO parameter values are used, under simplifying assumptions (Ozcan & Mohan 1999, Trelea 2003, Liu 2014, Bonyadi & Michalewicz 2016, Cleghorn & Engelbrecht 2018).

In the traditional sense, convergence is defined as (Cleghorn & Engelbrecht 2018)

Definition 3.1.1. *The sequence $(\mathbf{s}_t)$ in $\mathbb{R}^d$ is convergent if there exists an $\mathbf{s} \in \mathbb{R}^d$ such that*

$$\lim_{t \to \infty} \mathbf{s}_t = \mathbf{s} \tag{3.4}$$

This definition of deterministic convergence implies complete convergence in the sense that the particles will eventually stop moving. Due to the stochastic nature of PSO combined with the existence of two different locations of attraction (personal best and

global best locations) this type of convergence cannot be guaranteed. Instead, PSO stability analyses typically aim to guarantee order-1 and order-2 stability. Order-1 stability is defined as (Cleghorn & Engelbrecht 2018)

Definition 3.1.2. *The sequence* $(\mathbf{s}_t)$ *in* $\mathbb{R}^d$ *is order-1 stable if there exists an* $\mathbf{s}_E \in \mathbb{R}^d$ *such that*

$$\lim_{t\to\infty} E[\mathbf{s}_t] = \mathbf{s}_E \tag{3.5}$$

where $E[\mathbf{s}_t]$ *is the expected value of* $\mathbf{s}_t$.

On the other hand, order-2 stability is defined as (Cleghorn & Engelbrecht 2018)

Definition 3.1.3. *The sequence* $(\mathbf{s}_t)$ *in* $\mathbb{R}^d$ *is order-2 stable if there exists an* $\mathbf{s}_V \in \mathbb{R}^d$ *such that*

$$\lim_{t\to\infty} V[\mathbf{s}_t] = \mathbf{s}_V \tag{3.6}$$

where $V[\mathbf{s}_t]$ *is the variance of* $\mathbf{s}_t$.

In other words, order-1 and order-2 stability can be conceptually understood to imply that the particles will converge and oscillate around a location while the amplitude of the oscillations will eventually settle to a constant value (i.e. the particles will not diverge uncontrollably).

The aforementioned stability analyses showed that PSO is order-1 and order-2 stable, using simplifying assumptions. According to these studies, order-1 stability is guaranteed for PSO for the following parameter values

$$-1 < \omega < 1 \qquad 0 < \hat{c} < 4(\omega + 1), \tag{3.7}$$

where $\hat{c}$ is the *behaviour coefficient* and is given by

$$\hat{c} = c_1 + c_2. \tag{3.8}$$

On the other hand, order-2 stability is guaranteed by

$$-1 < \omega < 1 \qquad 0 < \hat{c} < \frac{24(1-\omega^2)}{7-5\omega}. \tag{3.9}$$

These *safe operating regions* are visualised in Figure 3-1.

Order-1 and order-2 stabilities are important parts of the original PSO algorithm because they provide a range of parameter values under which the algorithm remains stable. It is therefore crucial to show that any modifications made to the original PSO

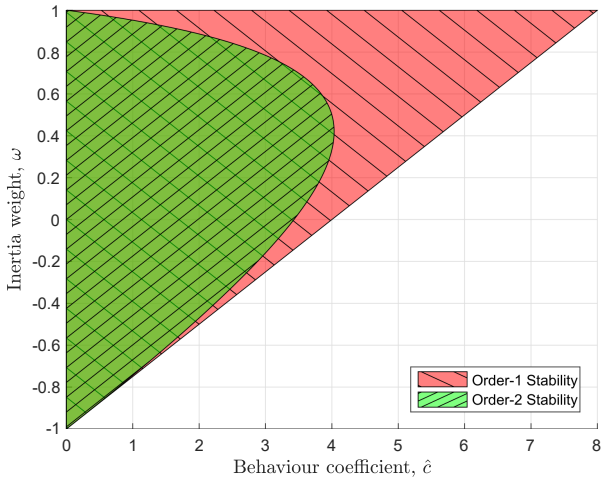


Figure 3-1: The *safe operating regions* defined by allowable values of $\omega, \hat{c}$ that guarantee order-1 and order-2 stability.

algorithm do not violate its order-1 and order-2 stabilities. With these restrictions in place, the next section will introduce modifications to address the problems of the original PSO discussed in the beginning of this chapter. The modifications will allow PSO to consider the maximum velocity and maximum acceleration limitations of the robots as well as the timestep size of the controller (i.e. controller refresh rate).

3.2 Adaptation of PSO for Swarm Robotics

The modifications introduced in this section will enable the PSO algorithm to consider the physical limitations of the robot (e.g. maximum velocity, maximum acceleration etc.). This can be achieved by re-defining the velocity and position update Equations (2.1) and (2.2).

First, a modified form of the position update Equation (2.2) is introduced,

$$\mathbf{x}[k+1] = \mathbf{x}[k] + \Delta t \ \mathbf{u}[k+1], \tag{3.10}$$

where Δt represents the discrete timestep of the PSO controller, which is the real-world time in seconds that corresponds to one PSO iteration. Delays caused by inter-robot communication and processing of sensor input will lead to larger values of Δt. The robot may employ other *low level* local controllers for tasks that require a higher refresh rate (e.g. motor controllers, data collection and data fusion controllers etc). The introduction of Δt allows the study of how different timestep sizes can affect the

operation of PSO, in order to address the second of the PSO problems outlined at the beginning of this chapter.

When it comes to Equation (2.1), the terms $(\mathbf{y}[k] - \mathbf{x}[k])$ and $(\mathbf{y_g}[k] - \mathbf{x}[k])$ are unconstrained, producing what was called Acceleration by Distance by Kennedy & Eberhart (1995). Due to these unconstrained terms, PSO does not have a maximum velocity and acceleration that can be equated to the maximum velocity and acceleration of the physical robots. Constraints can be introduced into these terms by replacing Equation (2.1) with

$$\mathbf{u}[k+1] = \omega\mathbf{u}[k] + c_1\mathbf{r_1} \circ \text{sgn}(\mathbf{y}[k] - \mathbf{x}[k]) + c_2\mathbf{r_2} \circ \text{sgn}(\mathbf{y_g}[k] - \mathbf{x}[k]), \tag{3.11}$$

where the sgn function is given by

$$\text{sgn}(x) = \left\{ \begin{array}{ll} 1, & x > 0 \\ 0, & x = 0 \\ -1, & x < 0 \end{array} \right\} \tag{3.12}$$

and it is used to limit the maximum acceleration caused by each accelerating term. Since the deceleration caused by the decelerating term $\omega\mathbf{u}[k]$ increases with velocity $\mathbf{u}[k]$, a point will be eventually reached where the acceleration caused by the accelerating terms will be cancelled out by the deceleration of $\omega\mathbf{u}[k]$ and therefore, maximum velocity will be achieved.

It should be noted that in the original PSO work, Kennedy & Eberhart (1995) start with this velocity update equation (implied through the use of algorithms), before switching to the currently known PSO velocity update equation of Equation (2.1). This is done in order to remove the maximum acceleration constraint from particles, allowing them to cover arbitrarily large distances - an ability that is desirable in parameter optimisation to reduce operational time. In swarm robotics on the other hand this is not desirable since the physical robots are not capable of covering arbitrarily large distances. Instead, the PSO velocity and acceleration limitations can be synchronised with the velocity and acceleration limitations of the physical robots, resulting in more accurate control. Therefore, this modification aims to address the first of the PSO problems outlined at the beginning of this chapter.

The resulting modified version of PSO will be referred to as the Adapted PSO algorithm for the rest of this thesis. With Adapted PSO introduced, it is important to identify the range of parameter values that will ensure order-1 and order-2 stability, which will be discussed in the next section.

3.2.1 Updated Parameter Stability Criteria

The stability criteria of Equation (3.7) and Equation (3.9) must now be redefined for Adapted PSO to ensure stability. It can be shown that the criteria for both order-1 and order-2 stability are

$$-1 < \omega < 1 \qquad \hat{c} > 0. \tag{3.13}$$

For proof see appendix A. Even though the criteria of Equation (3.13) ensure stability, using negative values of ω is nonsensical for swarm robotic applications, since it would result in the velocity of the robot changing direction at every timestep. Therefore, the rest of this thesis will consider the subset

$$0 \leq \omega < 1 \qquad \hat{c} > 0. \tag{3.14}$$

Apart from the parameter tuning criteria, one further conclusion can be made about the behaviour of Adapted PSO by observing Equation (3.11). The sgn function is not a smooth function and may result in what is known as chattering (Utkin 2011) under certain conditions. Chattering is often considered a harmful phenomenon characterised by oscillations of constant amplitude that can result in wear of moving mechanical parts and heat loses in electrical power circuits and it is described by the position and velocity equations (Utkin 2011)

$$\mathbf{x} = g(\mathbf{x}, t) + b\mathbf{u} \qquad g, b \in \mathbb{R}^d \tag{3.15}$$

and

$$\mathbf{u} = -L\text{sgn}(s(\mathbf{x})), \tag{3.16}$$

where $L > 0$ is a constant.

The Adapted PSO position update equation (3.10) is already in the same form as (3.15). When $\mathbf{u}[k] \approx 0$ and $\mathbf{y} \approx \mathbf{y_g}$, the Adapted PSO velocity update equation (3.11) takes the form

$$\mathbf{u}[k+1] \approx -\hat{c}\mathbf{r} \circ \text{sgn}(\mathbf{x}[k] - \mathbf{y_g}[k]), \tag{3.17}$$

which is a stochastic form of the velocity update chattering equation (3.16) (i.e. (3.16) with an additional random variable $\mathbf{r} \in [0, 1]$). These conditions generally occur after the swarm has converged to the source and therefore they do not affect the convergence behaviour of Adapted PSO or its stability. If chattering needs to be avoided, a smooth function can be used instead of the sgn function (e.g. tanh or some type of a logistic function with outputs in the range (-1,1)).

Taking tanh as an example, if $|(\mathbf{y}[k] - \mathbf{x}[k])| \gg 0$, then

$$\tanh(\mathbf{y}[k] - \mathbf{x}[k]) \approx \text{sgn}(\mathbf{y}[k] - \mathbf{x}[k]).$$

On the other hand, if $|(\mathbf{y}[k] - \mathbf{x}[k])| \approx 0$, then

$$\tanh(\mathbf{y}[k] - \mathbf{x}[k]) \approx \mathbf{y}[k] - \mathbf{x}[k].$$

In this way, when the swarm is far away from its personal and global best locations, where chattering is not a problem, the velocity update equation will behave like the Adapted PSO velocity update equation (3.11). On the other hand, as the swarm gets closer to convergence, where chattering can occur, the velocity update equation will behave like the original PSO velocity update equation (2.1), which is not affected by chattering. This thesis considers the sgn function for simplicity but all of the results of the following analysis can be applied to the aforementioned smooth functions as well.

Based on the criteria of Equation (3.14), the next section will present an analysis that will aim to describe how the velocity of an Adapted PSO particle may change at each timestep. The analysis will derive expressions of maximum velocity and maximum acceleration in terms of the inertia weight ω, the behaviour coefficient $\hat{c}$ and the timestep size Δt. In this way by properly tuning these parameters it will be possible to fully synchronise Adapted PSO with the motion of the physical robots, addressing the problems outlined at the beginning of this chapter.

3.3 Control of Velocity and Acceleration

The stability analyses of PSO described in Section 3.1.1 focus on studying the long term behaviour of PSO but they do not try to understand the timestep-to-timestep behaviour of the particles. This is because in parameter optimisation, practitioners are not concerned with the short-term behaviour of the particles but are instead interested about the final convergence location of the swarm at the end of the simulation.

In order to simplify the following analysis, and in line with assumptions made in previous analyses, $\mathbf{y_g}$ will be drawn from a distribution with well-defined mean μ and variance σ (Bonyadi & Michalewicz 2016, Cleghorn & Engelbrecht 2018). Note that since only a single particle is considered, the index i will be omitted.

First, assume that the personal best location $\mathbf{y}$ and the global best location $\mathbf{y_g}$ lie towards the same general direction relative to the particle. In these extreme conditions,

the accelerating terms act constructively (i.e. they do not cancel each other out) resulting in maximum acceleration, while in all other cases, the acceleration is not maximised. Under these conditions, Equation (3.11) can be simplified based on the stability study of Trelea (2003) so that

$$\mathbf{u}[k+1] = \omega\mathbf{u}[k] + \hat{c}\hat{\mathbf{r}} \circ \text{sgn}(\hat{\mathbf{y}}[k] - \mathbf{x}[k]), \tag{3.18}$$

where $\hat{\mathbf{y}}[k]$ is the weighted average of $\mathbf{y}[k]$ and $\mathbf{y}_g[k]$ as shown below

$$\hat{\mathbf{y}}[k] = \frac{c_1}{c_1 + c_2}\mathbf{y}[k] + \frac{c_2}{c_1 + c_2}\mathbf{y_g}[k].$$

In this way, the cognitive and social coefficients (c_1 and c_2) are replaced with the behaviour coefficient $\hat{c}$ of Equation (3.8). The vector $\hat{\mathbf{r}}$ is a random vector of which, each component r_j is drawn from the uniform distribution such that

$$\hat{r}_j \sim U(0,1), \quad 1 \leq j \leq d.$$

Building on Equations (3.10) and (3.18), the following analysis will consist of three stages: Firstly, a state model will be derived in matrix form that describes the particle's motion at each timestep, secondly the state model will be decomposed to understand how the state changes from one timestep to another, and finally expressions for the maximum velocity and acceleration of the particle will be derived in terms of ω and $\hat{c}$.

3.3.1 State model

This section will introduce a state model that will describe how the velocity and acceleration of a single particle vary at each timestep. In control theory, the *state* of a body may be described by its position and velocity at a specific timestep. The state vector $\mathbf{z}[k]$, which describes the state of motion of the particle in each of the d dimensions, is given by

$$\mathbf{z}[k] = \begin{bmatrix} \mathbf{z}_1[k] \\ \mathbf{z}_2[k] \\ \vdots \\ \mathbf{z}_d[k] \end{bmatrix} \tag{3.19}$$

where $\mathbf{z}_j[k]$ describes the state of motion in only a single dimension j such that

$$\mathbf{z}_j[k] = \begin{bmatrix} x_{\ j}[k] \\ u_{\ j}[k] \end{bmatrix}, \tag{3.20}$$

where $x_j[k]$ and $u_j[k]$ are the j^{th} components of position ($\mathbf{x}[k]$) and velocity ($\mathbf{u}[k]$). PSO algorithms have no interdependency between dimensions, therefore it is possible to use the single-dimension state vector $\mathbf{z}_j[k]$ as a general description of every dimension of $\mathbf{z}[k]$. Therefore, Equations (3.10) and (3.18) can be written in matrix form for each dimension j as

$$\mathbf{z}_j[k+1] = \mathbf{M}\mathbf{z}_j[k] \ + \mathbf{b}_j[k], \tag{3.21}$$

where, the right-hand-side consists of a deterministic term $\mathbf{M}\mathbf{z}_j[k]$ and a stochastic term $\mathbf{b}_j[k]$, such that

$$\mathbf{M} = \begin{bmatrix} 1 & \Delta t \\ 0 & \omega \end{bmatrix}, \quad \mathbf{b}_j[k] = \hat{c} \times \text{sgn}(\hat{y}_j[k] - x_j[k]) \begin{bmatrix} 0 \\ 1 \end{bmatrix} \times r_j.$$

This is a second order system in terms of position and velocity. However, as the objective is to understand how ω and $\hat{c}$ will affect velocity and acceleration, acceleration will be included as a state variable. By differentiating Equation (3.18), the acceleration is given by

$$\mathbf{a}[k+1] \ = \frac{(\omega - 1)\mathbf{u}[k] + \hat{c}\hat{\mathbf{r}} \circ \text{sgn}(\hat{\mathbf{y}}[k] - \mathbf{x}[k])}{\Delta t}. \tag{3.22}$$

Furthermore, since the analysis focuses on the timestep-to-timestep behaviour of the particles, position is of low importance and both the velocity and the acceleration are linearly independent of the position. Therefore for the sake of simplification position will be removed from the state. Thus, a new single-dimension state vector $\hat{\mathbf{z}}$ is defined as

$$\hat{\mathbf{z}}_j[k] = \begin{bmatrix} u_j[k] \\ a_j[k] \end{bmatrix}, \qquad 1 \leq j \leq d.$$

Using Equation (3.18) and Equation (3.22), the new state model is given by

$$\hat{\mathbf{z}}_j[k+1] = \hat{\mathbf{M}}\hat{\mathbf{z}}_j[k] \ + \hat{\mathbf{b}}_j[k], \tag{3.23}$$

where, the right-hand-side consists of a deterministic term $\hat{\mathbf{M}}\hat{\mathbf{z}}_j[k]$ and a stochastic term $\hat{\mathbf{b}}_j[k]$, such that

$$\hat{\mathbf{M}} = \begin{bmatrix} \omega & 0 \\ \frac{(\omega-1)}{\Delta t} & 0 \end{bmatrix}, \quad \hat{\mathbf{b}}_j[k] = \hat{c} \times \text{sgn}(\hat{y}_j[k] - x_j[k]) \begin{bmatrix} 1 \\ \frac{1}{\Delta t} \end{bmatrix} \times r_j.$$

This new state model can now be analysed using control theory and linear algebra techniques to determine the timestep-to-timestep behaviour of a single particle. The next stage will show how this can be done.

3.3.2 State Space Analysis

As explained in Section 3.3.1, Equation (3.23) describes only a single dimension j. This is because each dimension of PSO is decoupled from the rest (i.e. there is no interdependency between dimensions) and can be analysed individually. In the same manner, the following analysis will initially be performed on a single dimension j and at the end all dimensions will be combined to describe the behaviour of the full velocity and acceleration vectors.

Figures 3-2a to 3-2c are phase-space plots that describe the effect of the linear dependencies of the state model of Equation (3.23) (i.e. the deterministic term $\hat{\mathbf{M}}\hat{\mathbf{z}}_j[k]$) for $\omega = 0$, $0 \leq \omega < 1$ and $\omega = 1$ respectively. In other words, these plots show how the velocity and acceleration of the particle change due to the effect of the decelerating term $\omega \mathbf{u}_j[k]$ of Equation (3.18).

In these plots, each state vector (e.g. $\hat{\mathbf{z}}_j[k]$, $\hat{\mathbf{M}}\hat{\mathbf{z}}_j[k]$ etc.) describes a single point. The arrows in the phase-space plots show the direction of change from $\hat{\mathbf{z}}_j[k]$ to $\hat{\mathbf{M}}\hat{\mathbf{z}}_j[k]$ and the length of the arrows is proportional to the magnitude $|\hat{\mathbf{M}}\hat{\mathbf{z}}_j[k] - \hat{\mathbf{z}}_j[k]|$. The arrows in the phase-space plots show that the linear dependencies described by $\hat{\mathbf{M}}$ always cause the state vector $\hat{\mathbf{z}}_j$ to asymptotically converge towards the origin for $0 \leq \omega < 1$.

The phase-space plots show the direction towards $\hat{\mathbf{M}}\hat{\mathbf{z}}_j[k]$ but the do not show its exact location on the plots. It can be shown that $\hat{\mathbf{M}}\hat{\mathbf{z}}_j[k]$ always lies on the line

$$a_1(u, \omega, \Delta t) = \frac{\omega - 1}{\omega \Delta t} u, \tag{3.24}$$

(i.e. the dashed line shown in the plots). For proof, see Appendix B. Combining this information with the direction of the arrows in Figures 3-2a to 3-2c, it is possible to completely predict the location of $\hat{\mathbf{M}}\hat{\mathbf{z}}_j[k]$ for any $\hat{\mathbf{z}}_j[k]$, as shown in Figure 3-3a.

The location of $\hat{\mathbf{z}}_j[k+1]$ can be found by adding the stochastic term $\hat{\mathbf{b}}_j[k]$ to $\hat{\mathbf{M}}\hat{\mathbf{z}}_j[k]$ as shown in Equation (3.23). It is possible to show that $\hat{\mathbf{z}}_j[k+1]$ always lies in between the lines

$$\begin{aligned} a_2(u, \omega, \Delta t, \hat{c}) &= a_1(u - \hat{c}, \omega, \Delta t) + \frac{\hat{c}}{\Delta t} \\ a_3(u, \omega, \Delta t, \hat{c}) &= a_1(u + \hat{c}, \omega, \Delta t) - \frac{\hat{c}}{\Delta t}. \end{aligned} \tag{3.25}$$

For proof, see Appendix C. An example of this can be seen in Figure 3-3b, where the red arrow represents the vector $\hat{\mathbf{b}}_j[k]$ (i.e. the change from $\hat{\mathbf{M}}\hat{\mathbf{z}}_j[k]$ to $\hat{\mathbf{z}}_j[k+1]$).

To understand the figure more intuitively, note that the hatching lines of the green-

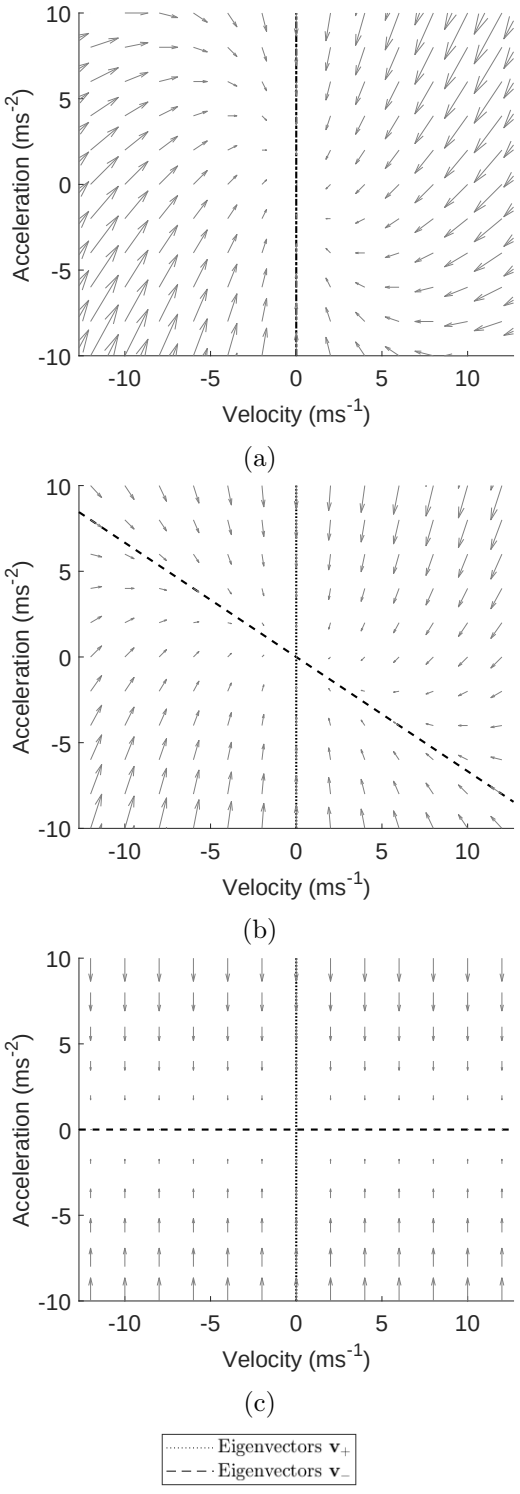


Figure 3-2: The phase-space graph for three different cases: (a) $\omega = 0$, (b) $0 \leq \omega < 1$ (in this specific case $\omega = 0.6$), (c) $\omega = 1$. In all three cases the system is stable. Also, in (a) (extreme case) and (b) (normal case), the system is asymptotically convergent towards the origin.

shaded region have gradient $\frac{1}{\Delta t}$ and that the red arrow is always parallel to them (as it can be also seen by the definition of $\hat{\mathbf{b}}_j[k]$ under Equation (3.23)). Additionally, $\hat{\mathbf{b}}_j[k]$ can have a maximum magnitude of $\hat{c}^2(1+\frac{1}{\Delta t^2})$, which can allow it to span the distance from the line a_1 to either of the lines a_2 and a_3. Therefore, $\hat{\mathbf{z}}_j[k+1]$ must always lie in between the lines a_2 and a_3.

The behaviours explained by Figures 3-3a and 3-3b describe every possible change in the velocity and acceleration of a particle controlled by the PSO velocity update equation Equation (3.11). Using these figures, it is now possible to derive expressions for the maximum velocity, maximum acceleration and maximum deceleration of the particle in terms of ω and $\hat{c}$.

3.3.3 Derivation of Extreme Cases

The purpose of this analysis was to identify the relationship between maximum velocity and acceleration and the values of ω and $\hat{c}$. It can be shown that there will always exist an asymptotically maximum velocity $U_j \geq 0$ given by

$$U_j = \frac{\hat{c}}{1-\omega} \tag{3.26}$$

such that $|u_j[k]| \leq U_j$. For proof see Appendix D. Figure 3-3b (green/shaded region) illustrates all of the possible locations of $\hat{\mathbf{z}}_j[k+1]$, when $-U_j \leq u_j[k] \leq U_j$. The stars ($\bigstar$) represent the locations with velocity U_j or $-U_j$. In these locations, the effect of the deterministic term $\hat{\mathbf{M}}\hat{\mathbf{z}}_j[k]$ (i.e. blue arrow in Figure 3-3a) may cancel out with the effect of the stochastic term $\hat{\mathbf{b}}_j$ (i.e. red arrow in Figure 3-3b when its magnitude is maximised), resulting in $\hat{\mathbf{z}}_j[k] = \hat{\mathbf{z}}_j[k+1]$.

Similarly, it can be shown that there will always exist a maximum acceleration $A_j^+ \geq 0$ given by

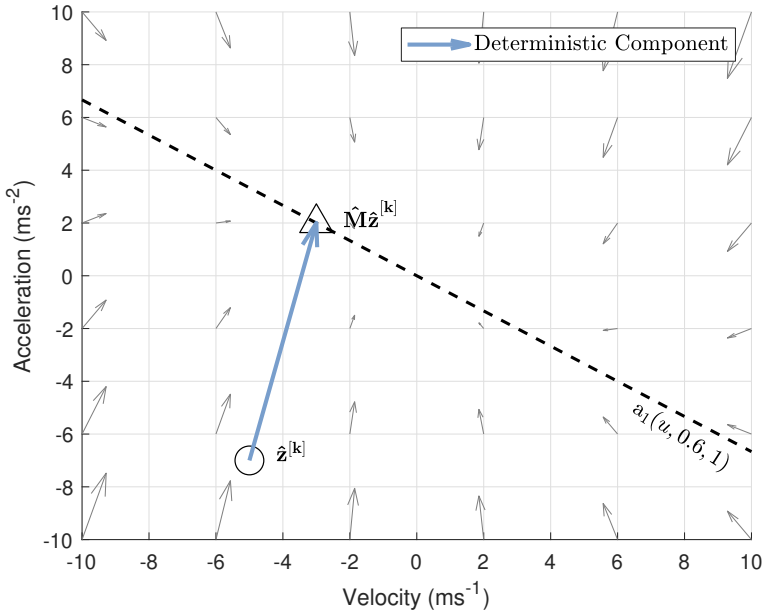
$$A_j^+ = \frac{\hat{c}}{\Delta t}, \tag{3.27}$$

and an asymptotically maximum deceleration $A_j^- \geq 0$ given by

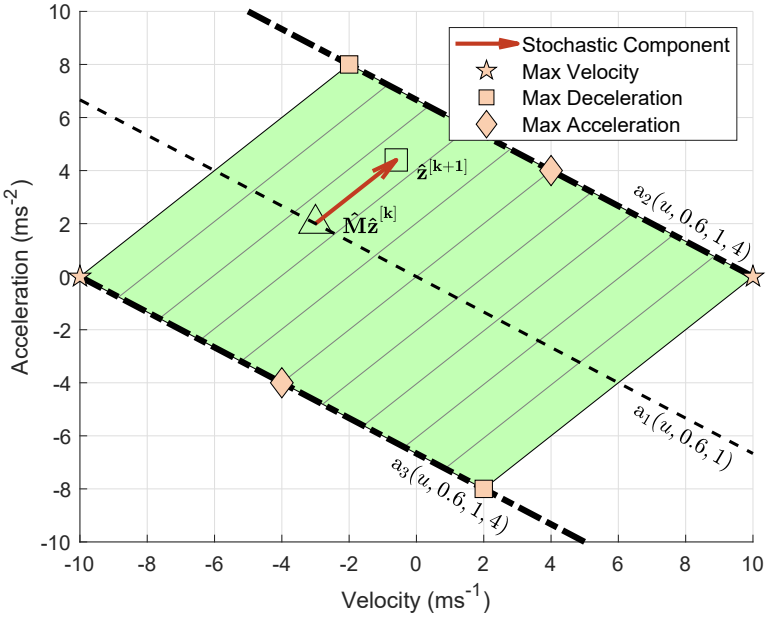
$$A_j^- = \frac{2\hat{c}}{\Delta t} = 2A_j^+. \tag{3.28}$$

For proof see Appendix E. The squares ($\blacksquare$) in Figure 3-3b represent the points of maximum deceleration A_j^- and the diamonds ($\blacklozenge$) the points of maximum acceleration A_j^+.

So far, the analysis was performed on an arbitrary single dimension j. Now, all dimen-



(a)



(b)

Figure 3-3: Example that shows the deterministic effect of $\hat{\mathbf{M}}$ (left) and the stochastic effect of $\hat{\mathbf{b}}_j[k]$ (right) on a random position of $\hat{\mathbf{z}}_j[k]$, for $\omega = 0.6$, $\hat{c} = 4$ and $\Delta t = 1$. No matter the location of $\hat{\mathbf{z}}_j[k]$, the point $\hat{\mathbf{M}}\hat{\mathbf{z}}_j[k]$ will always be located closer to the origin, lying on a_1. The point $\hat{\mathbf{z}}_j[k+1]$ will always be located in-between the lines a_2 and a_3. The vector $\hat{\mathbf{b}}_j[k]$ is always parallel to the hatching lines of the shaded-hatched region, which have gradient $\frac{1}{\Delta t}$. The shaded-hatched region represents all possible states $\hat{\mathbf{z}}_\mathbf{j}[k+1]$. For this system, $A_j^+ = 4\,\mathrm{m/s^2}$, $A_j^- = 8\,\mathrm{m/s^2}$ and $U_j = 10\,\mathrm{m/s}$.

sions can be combined to identify the overall maximum velocity U, maximum acceleration A^+ and maximum deceleration A^-. To find the maximum magnitude U of the velocity vector $\mathbf{u}$ it is necessary to equate each of its components to U_j, such that

$$U = U_j\sqrt{d} = \frac{\hat{c}\sqrt{d}}{1-\omega}. \tag{3.29}$$

Similarly, the maximum acceleration A^+ and the maximum deceleration A^- are given by

$$A^+ = A_j^+\sqrt{d} = \frac{\hat{c}\sqrt{d}}{\Delta t}, \tag{3.30}$$

$$A^- = A_j^-\sqrt{d} = \frac{2\hat{c}\sqrt{d}}{\Delta t} = 2A^+. \tag{3.31}$$

Finally, it is now possible to find expressions for the behaviour coefficient $\hat{c}$ and inertia weight ω in terms of A^+, A^-, U, d and Δt

$$\hat{c} = \frac{A^+(\Delta t)}{\sqrt{d}} = \frac{A^-(\Delta t)}{2\sqrt{d}}, \tag{3.32}$$

$$\omega = 1 - \frac{A^+(\Delta t)}{U} = 1 - \frac{A^-(\Delta t)}{2U}. \tag{3.33}$$

These expressions can be used to tune Adapted PSO based on the physical limitations of the robotic platform.

Equation Equation (3.33) can be difficult to use in its current form. To simplify it, a new parameter is introduced, called the sensitivity factor

$$\beta = \frac{A^+ \times \Delta t}{U} = \frac{A^- \times \Delta t}{2U} \tag{3.34}$$

such that

$$\omega = 1 - \beta. \tag{3.35}$$

Therefore, in order for $0 \leq \omega < 1$ to be satisfied, the sensitivity factor β must be always limited to $\beta \in (0, 1]$. From Equation (3.34) it can be also seen that if the maximum acceleration A^+ is too large compared to the maximum velocity U, the timestep Δt must be decreased (i.e. a faster controller must be used), in order to ensure that $\beta \in (0, 1]$ is satisfied. The sensitivity factor β is an important tool in Adapted PSO that can be used to describe several of its behaviours and it will be studied further in the simulations of Section 3.5. The next section will provide general tuning guidelines for Adapted PSO, based on the outcomes of the analysis of this section.

3.4 Guidelines

Section 3.3 showed how expressions of the inertia weight ω and behaviour coefficient $\hat{c}$ were derived, in terms of the maximum velocity U, the maximum acceleration and deceleration A^+ and A^-, the timestep size Δt and the number of dimensions of the environment d. This section will provide a set of design guidelines for the application of the Adapted PSO to swarm robotic tasks, using these expressions.

The parameter selection steps are as follows:

1. **Identify Δt:** The timestep size needs to be large enough to accommodate the time delay introduced by computationally expensive tasks and communications between robots.

2. **Identify U:** The desired maximum speed of the robot. It must be made sure that this does not exceed the actual maximum speed that the robot can achieve.

3. **Calculate either A^+ or A^-:** The desired maximum acceleration or deceleration using Equation (3.34). It must be made sure that they do not exceed the actual maximum acceleration or deceleration that the robot can achieve and that the sensitivity factor β is in the range (0,1].

4. **Calculate ω and $\hat{c}$:** Use Equation (3.35) and Equation (3.32) respectively.

5. **Ensure that ω and $\hat{c}$ satisfy the criteria of Equation (3.14):** If not, then a faster controller is required (i.e. smaller Δt)

6. **Select appropriate values for c_1 and c_2:** The sum of the accelerating coefficients is given by $\hat{c} = c_1 + c_2$.

Control of Exploration/Exploitation Tendencies: Traditional guidelines for PSO tuning in parameter optimisation tasks aim to control the convergence properties of the swarm (e.g. faster convergence, exploration/exploitation tendencies etc.). This is because in PSO the parameters are directly linked to the convergence behaviour of the swarm. In Adapted PSO however, the PSO parameters are primarily used to provide optimal control of the motion of robots. The guidelines provided ensure that the values of ω, c_1 and c_2 are properly tuned to provide the desired maximum velocity U and acceleration A^+, which can be associated to the physical maximum velocity and acceleration of the robot. Therefore, increasing the values of these parameters further will not result in faster convergence, and it can cause de-synchronisation between the PSO controller and the robot, resulting in poor control. Therefore, if faster convergence

is required, robots with higher maximum velocity and acceleration need to be used. The practitioner can still control the exploration/exploitation tendencies of the swarm, by adjusting the ratio $\frac{c_1}{c_2}$, like in the original PSO, but the limitations and guidelines provided in this chapter should also be followed.

Timestep Size Selection: Computationally intensive tasks may interfere with the operation of Adapted PSO. To avoid this, the value of Δt needs to be carefully selected. The timestep size Δt is a powerful feature of Adapted PSO that allows the robot to accurately predict its state in the next timestep. Therefore, as long as Δt accommodates all possible delays, it can ensure successful control of the swarm (by accommodating for the maximum velocity and acceleration of the physical robots). The next section will study further how different values of Δt can affect control.

Dynamic Velocity Control: Lastly, it should be noted that the desired maximum velocity U and desired maximum acceleration A^+ do not necessarily need to be equated to the actual maximum velocity and acceleration of the physical robot. Instead, they can have any value desired by the practitioner as long as this does not exceed the actual physical limitations of the robot. This implies another powerful feature of Adapted PSO that is not shared by most other PSO versions. The guidelines presented in this section can be used to dynamically control the maximum velocity and acceleration of the robot by re-tuning the PSO parameters at each timestep. Therefore, the robots can be forced to slow down when desired (e.g. in the presence of obstacles or when close to the source) and speed up when larger distances need to be covered. Dynamic Velocity Control (DVC) will be studied in the rest of this chapter through the use of simulations and more extensively in the following chapters.

3.5 Simulations

Two sets of simulations were carried out to study whether the behaviour of PSO particles matches the behaviour described by the analysis of Section 3.3. The simulations were run in MATLAB, using a 1D environment (i.e. the particles were only allowed to move in a single dimension). Since there is no inter-dependency between the dimension of PSO, the behaviour of the particles in a single dimension can be readily extended to any number of dimensions.

In the following simulations, all of the simulated particles were initialised around the origin and collisions were ignored to allow the particles to move freely. The fitness f of each location was given by its position on the x-axis (i.e. $f(\mathbf{x}) = x_1$) and the particles

were controlled by the Adapted PSO update Equations (3.10) and (3.11). This results in the swarm constantly moving towards the right indefinitely. The simulation time for all of the following simulations was 100 s, allowing enough time to clearly identify trends in the behaviours of the swarms.

The purpose of these simulations is to observe how the velocity of the particles changes for different values of U, A^+ and Δt. At each second, the velocities of all particles in the swarm are compared and the highest velocity is recorded. Note that the values of U, A^+ and Δt used in the simulations are chosen to clearly show the differences in the behaviours of the particles and they do not necessarily correspond to values that would be encountered in the real world.

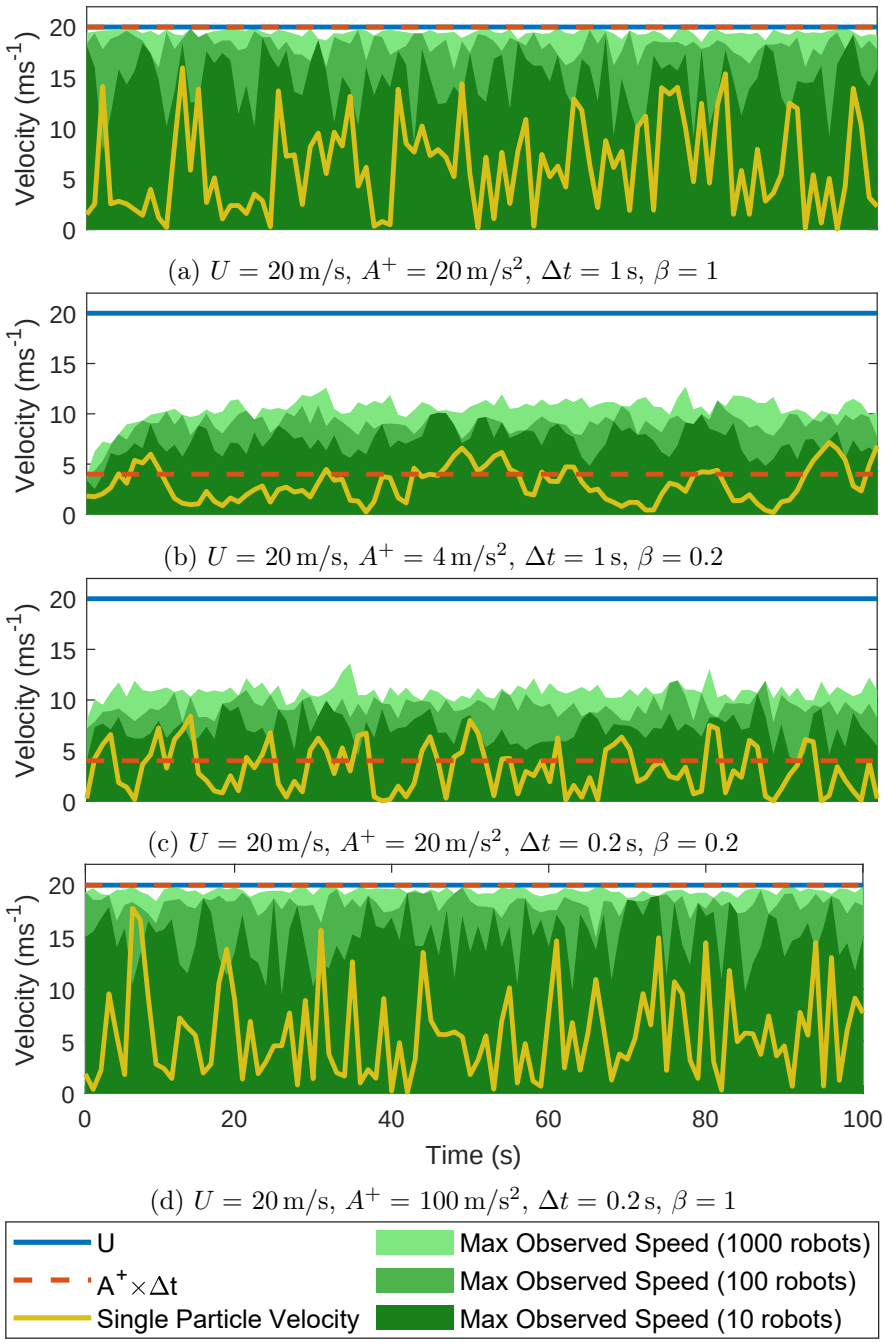
Since PSO is a stochastic algorithm (i.e. it makes use of random components), the velocity of the particles is rarely maximised. Therefore, the more particles that are included in the swarm, the more probable it becomes that one particle will be observed moving at a velocity close to the maximum. In order to understand the relationship between this probability and the number of particles, all of the following simulations are repeated with swarms of 10, 100 and 1000 particles. Furthermore, the velocity of a random single particle is also included to offer a visual understanding of the behaviour of individual particles.

3.5.1 Constant Parameters

The first set of simulations was performed for constant values of U, A^+ and Δt (throughout each simulation) resulting in constant PSO parameters ω and $\hat{c}$. Figure 3-4 shows the results of four different simulations run for different values of A^+ and Δt, where $U = 20\,\text{m/s}$. The sensitivity factor β (see Section 3.3.3) is also provided, to study how it can be used to describe the behaviour of the swarm.

In Figure 3-4a, $A^+ \times \Delta t = U$, resulting in sensitivity factor $\beta = 1$. The maximum observed velocities of all swarms appear to be close to the desired maximum velocity U but they never exceed it. The maximum velocity that is observed through the whole simulation is 19.9 m/s. This validates the conclusions of the analysis of Section 3.3, that U represents the maximum velocity that Adapted PSO can achieve.

In Figure 3-4b, the same simulations were repeated as in Figure 3-4a but the desired maximum acceleration A^+ is 5 times smaller. Similarly, in Figure 3-4c, the same simulations are repeated but in this case, the timestep size Δt is 5 times smaller. In both cases, $\beta = 0.2$. In these figures, the maximum velocities observed drop significantly (the maximum velocity observed through the whole simulation is 12.4 m/s for the former fig-



(a) $U = 20\,\mathrm{m/s}$, $A^+ = 20\,\mathrm{m/s^2}$, $\Delta t = 1\,\mathrm{s}$, $\beta = 1$

(b) $U = 20\,\mathrm{m/s}$, $A^+ = 4\,\mathrm{m/s^2}$, $\Delta t = 1\,\mathrm{s}$, $\beta = 0.2$

(c) $U = 20\,\mathrm{m/s}$, $A^+ = 20\,\mathrm{m/s^2}$, $\Delta t = 0.2\,\mathrm{s}$, $\beta = 0.2$

(d) $U = 20\,\mathrm{m/s}$, $A^+ = 100\,\mathrm{m/s^2}$, $\Delta t = 0.2\,\mathrm{s}$, $\beta = 1$

Figure 3-4: The maximum observed velocities at each second, for swarms of 10, 100 and 1000 particles (in order from darker green to lighter green regions). The blue solid line represents the value of the desired maximum velocity U and the red dashed line represents the value of the Acceleration-Time product ($A^+ \times \Delta t$. The yellow solid line is the velocity of a random particle. In (a) and (d), the simulations have the same sensitivity factor $\beta = 1$ and showcase similar maximum observed velocities. Similarly, in (b) and (c), the sensitivity factor is $\beta = 0.2$ and the simulations showcase similar maximum observed velocities.

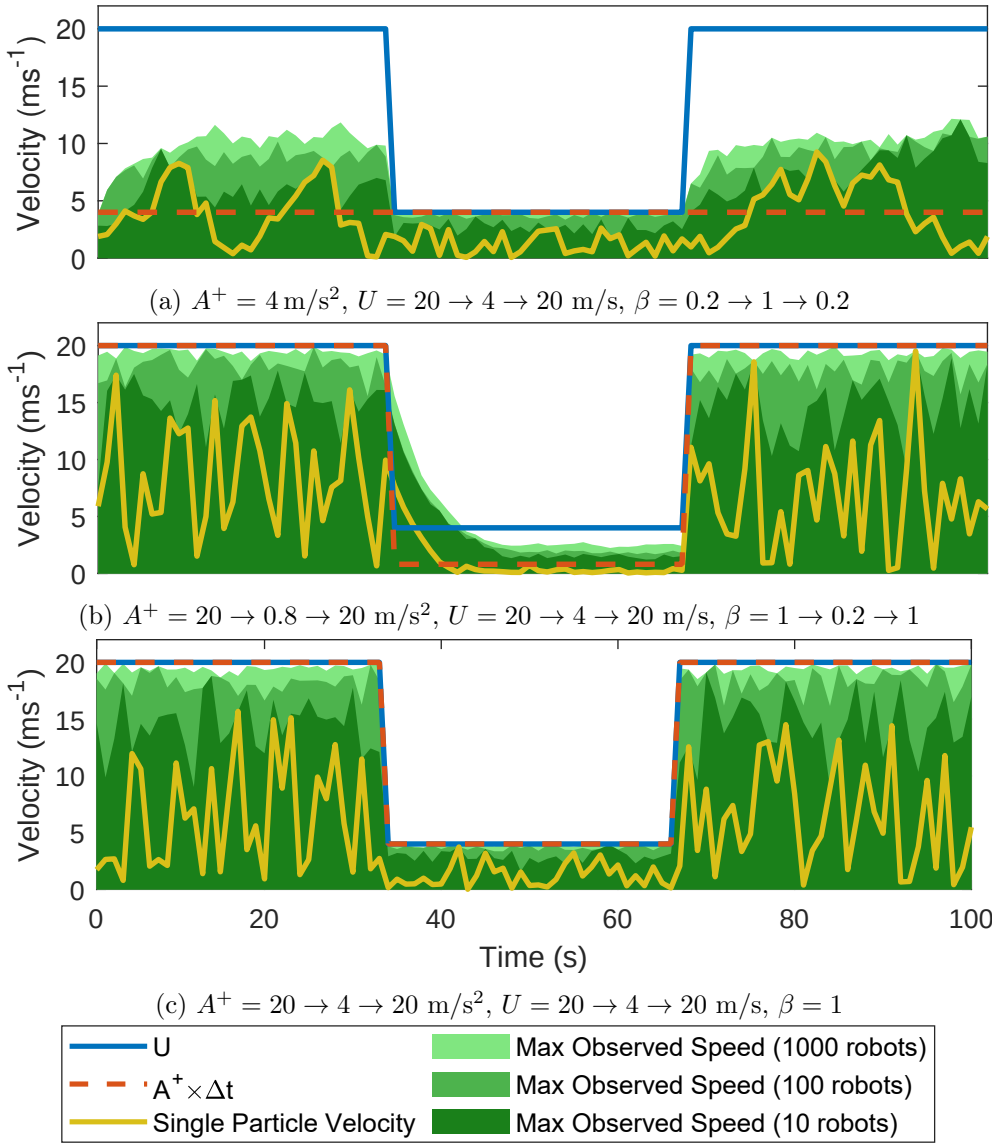
ure and 13.4 m/s for the latter). This can be intuitively understood by considering what happens as A^+ and Δt decrease. As these values drop, a particle requires more consecutive timesteps to reach maximum speed. Due to the stochastic nature of Adapted PSO, it is very rare that a particle will accelerate fully for several consecutive timesteps. This makes it more difficult to achieve maximum speed for smaller values of A^+ and Δt.

An interesting observation is that Figures 3-4b and 3-4c exhibit very similar observed maximum velocity behaviours. Furthermore, the two figures share the same Acceleration-Time product ($A^+ \times \Delta t$) and therefore the same sensitivity factor β. Therefore it can be hypothesised that the value of β indicates how likely it is that a particle will manage to reach maximum velocity. To test this, the fourth simulation was run with five times smaller Δt than in Figure 3-4a but with five times larger A^+, resulting in the same value of $\beta = 1$. The results of Figure 3-4d show that indeed the particles manage to reach maximum velocity, suggesting that the value of β can be used to predict the behaviour of the swarm (i.e. its ability to maximise its velocity).

3.5.2 Variable Parameters

As explained in Section 3.4, Adapted PSO offers the capability to dynamically control the maximum velocity of the particles through the process of DVC, by re-tuning the inertia weight ω and the accelerating coefficients c_1 and c_2 during the operation of the swarm. The second set of simulations ran, aimed to study how the Adapted PSO particles respond to such changes. The following simulations are identical to the ones presented in Figure 3-4 but the values of U and A^+ were altered at 34 s and 67 s, to study the step response of the swarm when the particle velocities need to either increase or decrease. Furthermore, different simulations aim to alter the value of the sensitivity factor β in different ways to observe how this affects the response of the swarm. The value of the timestep size Δt was 1 s for all simulations and the results are shown in Figure 3-5.

In Figure 3-5a, the desired maximum acceleration A^+ is constant at 4 m/s^2 and the desired maximum velocity U varies. During the first and third time intervals (0-33 s and 67-100 s), U is at 20 m/s, resulting in a sensitivity factor $\beta = 0.2$. During the second time interval (34-66 s), U drops to 4 m/s, resulting in $\beta = 1$. In the figure, it can be seen that when U drops at 34 s (high β), all of the maximum observed speeds follow immediately, resulting in a critically damped response. On the other hand, when U increases at 67 s (low β), the maximum observed speeds follow more slowly (it takes 6 seconds for the maximum observed speed of 1000 robots to exceed 9.5 m/s - the



(a) $A^+ = 4\,\mathrm{m/s^2}$, $U = 20 \to 4 \to 20$ m/s, $\beta = 0.2 \to 1 \to 0.2$

(b) $A^+ = 20 \to 0.8 \to 20$ m/s^2, $U = 20 \to 4 \to 20$ m/s, $\beta = 1 \to 0.2 \to 1$

(c) $A^+ = 20 \to 4 \to 20$ m/s^2, $U = 20 \to 4 \to 20$ m/s, $\beta = 1$

Figure 3-5: The maximum observed velocities at each second, for swarms of 10, 100 and 1000 particles (in order from darker green to lighter green regions). The blue solid line represents the value of the desired maximum velocity U and the red dashed line represents the value of the Acceleration-Time product ($A^+ \times \Delta t$. The yellow solid line is the velocity of a random particle. In (a), the desired maximum velocity U decreases at 34 s and increases again at 67 s varying β from low to high to low. In (b), both the desired maximum velocity U and acceleration A^+ decrease at 34 s and increase again at 67 s varying β from high to low to high. In (c), both the desired maximum velocity U and acceleration A^+ decrease at 34 s and increase again at 67 s while β remains high at all times.

lowest maximum observed speed in Figures 3-4b and 3-4c), resulting in an overdamped response. The same behaviour can be also seen at the beginning of the simulation, where it takes 8 seconds for the maximum observed speed of 1000 robots to exceed 9.5 m/s.

In Figure 3-5b, both A^+ and U vary in such a way that in the first and third time intervals, $\beta = 1$ and in the second time interval, $\beta = 0.2$. In this case, when U and A^+ drop at 34 s (low β), the overdamped response of the system is much more clear (this is due to the large velocity change demanded), where it takes 11 seconds for the maximum observed speed of 1000 robots to fall below 3 m/s. Conversely, when U and A^+ rise at 67 s (high β), the system response is critically damped.

Finally, in Figure 3-5c, both A^+ and U vary in such a way that the sensitivity factor $\beta = 1$ throughout the whole simulation. In this case, the response is critically damped, both at 34 s and at 67 s. The following section will discuss the results of the simulations presented in this section.

3.6 Discussion

The results presented in Section 3.5 show that the individual values of U, A^+ and Δt do not provide enough information about the behaviour of the Adapted PSO algorithm on their own. Instead, it is possible to better predict the behaviour of the algorithm by considering the sensitivity factor β. The results show that a high value of β allows the Adapted PSO to maximise its velocity (both average and maximum) and respond quickly to changes in the values of the desired maximum velocity U and acceleration A^+.

Since U and A^+ are limited by the physical limitations of the robot, the conclusions of these simulations primarily concern the selection of a timestep size Δt. In other words, the timestep size Δt should be selected by the practitioner to maximise the sensitivity factor β. As the value of Δt decreases (i.e. faster controller), the sensitivity factor β decreases as well, which could result in sub-optimal operation of Adapted PSO, both in terms of average and maximum velocity output and control. At first this may seem counter-intuitive, since faster controllers are typically associated with better system control (faster response to errors). That said, it should be understood that the aim of the simulations presented in this section was to study how the Adapted PSO controller responds to the values of U, A^+ and Δt (i.e. how the velocities outputted by the controller vary) and not how the physical robots responds to them.

To explain this further, consider the example of a robot with relatively low maximum acceleration compared to its maximum velocity (e.g. a vehicle moving on ice). In this case, if a small Δt is selected, the controller will quickly update its velocity outputs, before the robot manages to respond to them in any significant way. If a large Δt is used instead, it can allow enough time for the robot to accelerate towards a specific direction building up significant speed before the controller issues a new command. Alternatively, a large Δt value that was specifically selected to optimise β, allows the controller to know how quickly the robot will respond to its commands. This information can then be used to issue better timed commands, improving control.

In other words, Adapted PSO does not just consider the physical limitations of the robot, in terms of maximum velocity and acceleration output; it also considers this information to optimise the frequency that new commands are being issued, thereby synchronising further the controller with the physical platform. This way of employing Δt and β, along with the DVC process to achieve optimal motion control of a robot will be studied further in the following chapter.

3.7 Generalised Adapted PSO

Section 3.5 introduced a number of different useful properties and characteristics of the Adapted PSO that were derived from the analysis presented in this chapter. Apart from these, it is possible to recognise one further extension of the Adapted PSO that is implied by the presented analysis.

So far, Adapted PSO made use of two accelerating terms (i.e. cognitive and social), primarily because these terms are used by original PSO. The analysis merges these terms by adding their corresponding accelerating coefficients (c_1 and c_2) to form the behaviour coefficient $\hat{c}$. From there, all conclusions and resulting expressions are derived in terms of $\hat{c}$.

That said, there is no reason that Adapted PSO be limited to only two accelerating terms. Instead, the analysis presented can be applied to a more general velocity update equation that has an arbitrary number of n accelerating terms, resulting in the velocity update equation of the Generalised Adapted PSO

$$\begin{aligned}\mathbf{u}[k+1] = \omega\mathbf{u}[k] &+ c_1\mathbf{r_1} \circ \text{sgn}(\mathbf{y_1}[k] - \mathbf{x}[k]) \\ &+ c_2\mathbf{r_2} \circ \text{sgn}(\mathbf{y_2}[k] - \mathbf{x}[k]) \\ &+ ... + c_n\mathbf{r_n} \circ \text{sgn}(\mathbf{y_n}[k] - \mathbf{x}[k]), \quad (3.36)\end{aligned}$$

where $\mathbf{y_1}$, $\mathbf{y_2}$, ..., $\mathbf{y_n}$ are locations in the real world and

$$r_{1,j},\ r_{2,j},\ \ldots\ ,\ r_{n,j} \sim U(0,1) \quad 1 \leq j \leq d. \tag{3.37}$$

In a similar manner to Equation (3.8), the behaviour coefficient can be calculated by adding all accelerating coefficients

$$\hat{c} = c_1 + c_2 + ... + c_n, \tag{3.38}$$

Lastly, if it is assumed that all locations $\mathbf{y_1}$, $\mathbf{y_2}$, ..., $\mathbf{y_n}$ lie towards the same general direction relative to the particle, thereby maximising acceleration, Equation (3.36) can take the form of Equation (3.18), where $\hat{\mathbf{y}}[k]$ is now given by

$$\hat{\mathbf{y}}[k] = \frac{c_1}{c_1 + c_2 + ... + c_n}\mathbf{y_1}[k] + \frac{c_2}{c_1 + c_2 + ... + c_n}\mathbf{y_2}[k] + ... + \frac{c_n}{c_1 + c_2 + ... + c_n}\mathbf{y_n}[k].$$

From here, it is now possible to apply the analysis presented in this chapter, starting from Section 3.3, to the Generalised Adapted PSO algorithm, including all properties and behavioural characteristics described in Section 3.5.

The additional accelerating terms of Generalised Adapted PSO can be used to incorporate additional tasks into Adapted PSO such as obstacle avoidance, aggregation, flocking and target trapping etc. This can be achieved by replacing the input of the sgn function in each term, with the directional output (e.g. a virtual force or a trajectory direction) of other techniques and algorithms, designed to carry out specific individual tasks. Chapter 4 will show how this idea, combined with DVC can be used to incorporate obstacle avoidance and aggregation into Adapted PSO.

3.8 Conclusions

This chapter has introduced a modified version of the original PSO algorithm called Adapted PSO, that introduces maximum velocity and maximum acceleration limitations. It is shown that Adapted PSO is order-1 and order-2 stable, which are necessary requirements imposed by the literature. A timestep-to-timestep analysis is then presented that concludes with the derivation of expressions for the maximum velocity and maximum acceleration in terms of the inertia weight ω, the cognitive and social coefficients c_1 and c_2 and the timestep size Δt. In this way, Adapted PSO considers the physical limitations of the robots, addressing the first problem of PSO (see Section 3.1), while also considering the refresh rate of the controller, addressing the second problem.

Simulations were carried out to study the resulting behaviour of Adapted PSO and show that the behaviour in terms of maximum and average velocity output and response to input changes can be accurately described by the value of the sensitivity factor β. Lastly, a generalised extension of the algorithm called Generalised Adapted PSO is introduced, which can allow the incorporation of additional tasks into Adapted PSO, such as obstacle avoidance.

References

Bonyadi, M. R. & Michalewicz, Z. (2016), 'Stability Analysis of the Particle Swarm Optimization Without Stagnation Assumption', *IEEE Transactions on Evolutionary Computation* **20**(5), 814–819.

Cleghorn, C. W. & Engelbrecht, A. P. (2018), 'Particle swarm stability: a theoretical extension using the non-stagnate distribution assumption', *Swarm Intelligence* **12**(1), 1–22.

Clerc, M. & Kennedy, J. (2002), 'The particle swarm - explosion, stability, and convergence in a multidimensional complex space', *IEEE Transactions on Evolutionary Computation* **6**(1), 58–73.

Hereford, J. M., Siebold, M. & Nichols, S. (2007), Using the Particle Swarm Optimization Algorithm for Robotic Search Applications, *in* '2007 IEEE Swarm Intelligence Symposium', pp. 53–59.

Kennedy, J. (2010), 'Particle swarm optimization', *Encyclopedia of machine learning* pp. 760–766.

Kennedy, J. & Eberhart, R. (1995), Particle swarm optimization, *in* 'Proceedings of ICNN'95 - International Conference on Neural Networks', Vol. 4, pp. 1942–1948 vol.4.

Krishnanand, K. N. & Ghose, D. (2009), 'Glowworm swarm optimization for simultaneous capture of multiple local optima of multimodal functions', *Swarm Intelligence* **3**(2), 87–124.

Liu, Q. (2014), 'Order-2 Stability Analysis of Particle Swarm Optimization', *Evolutionary computation* **23**.

Ozcan, E. & Mohan, C. K. (1999), Particle swarm optimization: surfing the waves, *in* 'Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)', Vol. 3, pp. 1939–1944 Vol. 3.

Pugh, J. & Martinoli, A. (2007), Inspiring and Modeling Multi-Robot Search with Particle Swarm Optimization, *in* '2007 IEEE Swarm Intelligence Symposium', pp. 332–339.

Shi, Y. & Eberhart, R. C. (1998), Parameter selection in particle swarm optimization, *in* 'Evolutionary Programming VII', Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 591–600.

Trelea, I. C. (2003), 'The particle swarm optimization algorithm: convergence analysis and parameter selection', *Information Processing Letters* **85**(6), 317–325.

Utkin, V. (2011), 'Chattering Problem', *IFAC Proceedings Volumes* **44**(1), 13374–13379. **URL:** *https://www.sciencedirect.com/science/article/pii/S147466701645771X*

Chapter 4

Obstacle Avoidance and Dynamic Velocity Control Strategies

Chapter 3 introduced Adapted PSO, a variant of the original PSO algorithm that allows consideration of the physical movement limitations of the individual robots like maximum velocity and maximum acceleration. The chapter also introduced the concept of Dynamic Velocity Control (DVC). Lastly, a generalised version of the algorithm was introduced, called Generalised Adapted PSO, which is described by the following velocity update equation

$$\begin{aligned}\mathbf{u}[k+1] = \omega\mathbf{u}[k] &+ c_1\mathbf{r_1} \circ \text{sgn}(\mathbf{y_1}[k] - \mathbf{x}[k]) \\ &+ c_2\mathbf{r_2} \circ \text{sgn}(\mathbf{y_2}[k] - \mathbf{x}[k]) \\ &+ ... + c_n\mathbf{r_n} \circ \text{sgn}(\mathbf{y_n}[k] - \mathbf{x}[k]). \quad (4.1)\end{aligned}$$

where $\mathbf{y_1}$, $\mathbf{y_2}$, ..., $\mathbf{y_n}$ are locations in the real world.

While original and Adapted PSO make use of only *two* accelerating terms, Generalised Adapted PSO allows the use of n terms. These terms can be utilised to incorporate additional functionality such as obstacle avoidance, aggregation and flocking behaviours. This chapter will show how these concepts can be used to create a low-level motion controller for the control of robotic swarms.

4.1 Robotic Particle Swarm Optimisation

The idea of adding additional terms into the PSO velocity update equation to incorporate additional tasks into the algorithm has been previously explored in the literature, for example in Robotic Particle Swarm Optimisation (RPSO) (Couceiro et al. 2011).

RPSO aims to incorporate obstacle avoidance into PSO, by adding a third accelerating term. Therefore, the original PSO velocity update equation takes the following form

$$\mathbf{u}[k+1] = \omega\mathbf{u}[k] + c_1\mathbf{r_1} \circ (\mathbf{y}[k] - \mathbf{x}[k]) + c_2\mathbf{r_2} \circ (\mathbf{y_g}[k] - \mathbf{x}[k]) + c_3\mathbf{r_3} \circ (\mathbf{F_t}[k]) \quad (4.2)$$

where c_3 is the obstacle susceptibility coefficient and each element of $\mathbf{r_3}$ is drawn from the uniform distribution $U(0,1)$. Note that Equation (4.2) does not make use of a sgn function and therefore it does not inherit the capabilities of Generalised Adapted PSO presented in Chapter 3 (e.g. maximum velocity and acceleration).

The vector $\mathbf{F_t}$ is described as the direction of motion that optimises a monotonically decreasing *sensing function* (e.g. a function that depends on the output signal of proximity sensors). In this way, the third term always tries to move the particle away from nearby obstacles. A possible alternative definition of $\mathbf{F_t}$ is that of a *virtual force* used to push the robot away from surrounding obstacles, given by

$$\mathbf{F_t} = \sum_{p=1}^{P} \mathbf{F}_p, \quad (4.3)$$

where P is the total number of obstacles around the robot and $\mathbf{F}_p$ is a virtual repulsive force exerted by obstacle $p \in \{1, \dots, P\}$. Virtual forces are well-studied in swarm robotics and are generally used to control the position of a robot relative to its direct surroundings (e.g., obstacles, targets or other robots). There exist many different ways of defining a virtual force, ranging from simple position-dependent functions to more complex functions that are typically inspired from real-world physical forces (e.g., spring-damper systems) (Spears et al. 2004, Khaldi & Cherif 2016, Garone et al. 2018, Hosseinzadeh & Garone 2020). This chapter will make use of a simple position-dependent virtual force definition, but more complex definitions could also be used. Let $\mathbf{s}$ be a vector of distances between the robot and surrounding obstacles, such that, at $s_p = 0$ the robot has collided with obstacle p. A virtual repulsive force $\mathbf{F}_p$ exerted by obstacle p on robot i is defined as

$$\mathbf{F}_p = \frac{\gamma}{s_p} \frac{(\mathbf{x}_i - \mathbf{x}_p)}{|\mathbf{x}_i - \mathbf{x}_p|} \quad (4.4)$$

where $\mathbf{x}_i$ is the position of the centre of the robot and $\mathbf{x}_p$ is the position of the centre of obstacle p. The parameter $\gamma > 0$ is the avoidance sensitivity parameter and adjusts the intensity of the repulsive force. Note that (4.4) is undefined for $s_p = 0$ and $\mathbf{x}_i - \mathbf{x}_p = 0$. Nevertheless, such conditions occur only after a collision has happened and therefore they are assumed to not affect the operation of the system in practical applications.

Much like other modified versions of PSO, RPSO can be used as a trajectory planning algorithm, indicating the direction of motion of the robot. Therefore, the individual values of c_1, c_2 and c_3 do not matter, since the resulting velocity will typically exceed the maximum velocity of the robot. Instead, Couceiro et al. (2011) focused on the relationship between c_3 and $\max\{c_1, c_2\}$. When $c_3 \ll \max\{c_1, c_2\}$, the effect of the last term of Equation (4.2) may not be sufficient to force the robot to avoid an obstacle and if the cognitive and social terms happen to point towards the obstacle, the robot will collide with it. On the other hand, when $c_3 \gg \max\{c_1, c_2\}$, the robot becomes overly sensitive to obstacles. In this case, the obstacle susceptibility term overshadows the other terms, making it difficult for the robot to pass through small openings towards the source. Furthermore, all three accelerating terms are unbounded (i.e. a sgn function is not used) and can increase arbitrarily, leading to further collisions.

Couceiro et al. (2011) proposed that the value of c_3 could be dynamically recalibrated at each timestep, in a similar manner to the concept of DVC, seemingly in an attempt to improve the aforementioned RPSO behaviour. When no obstacles are present, c_3 tends to 0, but it grows the closer the robot is to an obstacle. However, the unbounded nature of the accelerating terms of RPSO means that each term can grow arbitrarily, no matter what the value of its accelerating coefficient is. For example, even if c_3 is large when close to an obstacle, the effect of the other accelerating terms may still end up being larger (if the personal and global best locations are far away from the robot), resulting in collisions. This weakness of RPSO can be explained mathematically as the accelerating terms having no expected magnitude. Therefore, to achieve control of the accelerating terms through the calibration of the accelerating coefficients, it is important to adapt the accelerating terms so that they have well-defined expected magnitude that depends on their corresponding accelerating coefficients. In Generalised Adapted RPSO, each arbitrary n^{th} accelerating term has an expected magnitude of $\frac{c_n}{2}$ and therefore, it can be used to overcome this weakness of RPSO. The rest of this chapter will show how this can be done.

4.2 Adapted Robotic Particle Swarm Optimisation

Generalised Adapted PSO can take a similar structure to the RPSO algorithm of Equation (4.2) by using three accelerating terms, resulting in the Adapted RPSO algorithm

$$\mathbf{u}[k+1] = \omega\mathbf{u}[k] + c_1\mathbf{r_1} \circ \text{sgn}(\mathbf{y}[k] - \mathbf{x}[k]) + c_2\mathbf{r_2} \circ \text{sgn}(\mathbf{y_g}[k] - \mathbf{x}[k]) + c_3\mathbf{r_3} \circ \text{sgn}(\mathbf{F_t}[k]). \quad (4.5)$$

The position update equation is the same as in Adapted PSO of Chapter 3

$$\mathbf{x}[k+1] = \mathbf{x}[k] + \Delta t\ \mathbf{u}[k+1], \quad (4.6)$$

In contrast to Equation (4.2), Adapted RPSO makes use of a sgn function in each accelerating term. Therefore, it inherits all of the characteristics of Generalised Adapted PSO presented in Chapter 3, such as a desired maximum velocity U, a desired maximum acceleration A^+ and a desired maximum deceleration $A^- = 2A^+$ given by

$$U = \frac{\hat{c}\sqrt{d}}{1-\omega} \qquad A^+ = \frac{\hat{c}\sqrt{d}}{\Delta t} \qquad A^- = \frac{2\hat{c}\sqrt{d}}{\Delta t}. \quad (4.7)$$

where Δt is the timestep size and $\hat{c} = c_1 + c_2 + c_3$. A sensitivity factor $0 < \beta \leq 1$ is also given by

$$\beta = \frac{A^+ \times \Delta t}{U} = \frac{A^- \times \Delta t}{2U}, \quad (4.8)$$

and therefore, the parameters ω and $\hat{c}$ are given by

$$\omega = 1 - \beta \qquad \hat{c} = \frac{A^+(\Delta t)}{\sqrt{d}}. \quad (4.9)$$

Furthermore, the parameter selection guidelines presented in Chapter 3 also apply to it. As a reminder, these guidelines are presented again for Adapted RPSO:

1. **Identify Δt:** The timestep size needs to be large enough to accommodate the time delay introduced by computationally expensive tasks and communications between robots.

2. **Identify U:** The desired maximum speed of the robot. It must be made sure that this does not exceed the actual maximum speed that the robot can achieve.

3. **Calculate either A^+ or A^-:** The desired maximum acceleration or deceleration using Equation (4.8). It must be made sure that they do not exceed the actual maximum acceleration or deceleration that the robot can achieve and that the

sensitivity factor β is in the range (0,1]. Ideally, β should be as high as possible as concluded from the results of Chapter 3.

4. **Calculate ω and $\hat{c}$:** Use Equation (4.9).

5. **Select desired values for c_1, c_2 and c_3:** The sum of the accelerating coefficients is given by $\hat{c} = c_1 + c_2 + c_3$.

4.2.1 Calibration of Individual Accelerating Terms

The last step of the tuning guidelines specifies that the sum of the accelerating coefficients should satisfy $\hat{c} = c_1 + c_2 + c_3$, but a way to select the value of each individual coefficient is not provided. As described in Section 4.1, the coefficients could be calibrated in such a way that c_3 increases and c_1 and c_2 decrease as the robot gets closer to the closest obstacle and vice versa as the robot moves away from it. That said, it is still unclear how much the increase or decrease should be depending on the distance to the obstacle.

This can be problematic, since if c_1 and c_2 are not sufficiently decreased when the robot is close to an obstacle, they may end up overshadowing the obstacle susceptibility term (third accelerating term), resulting in collisions. Furthermore, due to the stochastic component $\mathbf{r}$ of each term, the obstacle susceptibility term may end up being very small, even if c_3 is large, resulting in further collisions. Therefore, it is needed to provide a relationship between the distance to an obstacle and the new values of the accelerating coefficients that will ensure effective collision avoidance.

According to Equation (4.7), the maximum acceleration of Adapted RPSO is provided by

$$A^+ = \frac{(c_1 + c_2 + c_3)\sqrt{d}}{\Delta t}. \tag{4.10}$$

Equation (4.10) is linear and therefore the maximum acceleration that can be caused by each accelerating term can be calculated using a weighted average of the corresponding accelerating coefficient, such that

$$A_1^+ = \frac{c_1 \times \sqrt{d}}{\Delta t} \qquad A_2^+ = \frac{c_2 \times \sqrt{d}}{\Delta t} \qquad A_3^+ = \frac{c_3 \times \sqrt{d}}{\Delta t}. \tag{4.11}$$

Similarly, based on Equation (4.8), the maximum velocity that can be caused by each accelerating term is given by

$$U_1 = \frac{A_1^+ \times \Delta t}{\beta} \qquad U_2 = \frac{A_2^+ \times \Delta t}{\beta} \qquad U_3 = \frac{A_3^+ \times \Delta t}{\beta}. \tag{4.12}$$

This concept can be extended to any combination of accelerating terms, by setting $\hat{c}$ equal to the sum of their corresponding accelerating coefficients in Equations (4.7) and (4.8). Therefore, it is now possible to use the maximum velocity caused by any combination of terms to limit them accordingly. For example the cognitive and social coefficients (c_1 and c_2) can always be limited such that the maximum velocity caused by them will never be enough to cause a collision with the closest obstacle, even if the obstacle susceptibility term happens to be small at that point.

This concept can be used to design a DVC strategy that will control the value of each accelerating coefficient at each timestep. The next section will describe how Adapted RPSO combined with a properly designed DVC strategy can be used to overcome the limitations of original RPSO, discussed in Section 4.1.

4.2.2 Implementation of Dynamic Velocity Control Strategy

The following DVC strategy will aim to dynamically adjust the RPSO parameters at each timestep to control the maximum velocity of each robot. The parameters are adjusted so that convergence to the personal and global best locations is prioritised when the robot is far away from obstacles, while obstacle avoidance is prioritised when an obstacle is nearby. When implemented correctly, this can prevent collisions with obstacles and other agents, while also achieving convergence of the swarm to the source.

Since Generalised Adapted PSO is a modified version of the original PSO, it inherits its scalability characteristics. That said, it is important that the DVC strategy is designed in such a way as to ensure that the overall control process remains scalable. To achieve this, the DVC strategy is designed to ensure that a single robot will be able, at any point, to avoid collision with its surroundings (both obstacles and other robots), while still being able to converge towards its personal best and global best locations. As long as this is true, the DVC strategy will be scalable to any number of robots. The rest of this section will describe the DVC strategy used.

Let $\mathbf{s}$ be the list of distances to the nearest obstacles, sorted in order from smallest to largest, such that s_1 is the distance to the closest obstacle. Similarly, let

$$\boldsymbol{\nu} = \frac{\mathbf{s}}{\Delta t}, \tag{4.13}$$

be a vector of speeds, such that ν_1 is the minimum speed required for the robot to collide with the closest obstacle in the next timestep. The value ν_p can be used to prevent collision with obstacle p by selecting a desired maximum speed U smaller than

ν_p using

$$U = \alpha \times \nu_p, \tag{4.14}$$

where $0 < \alpha < 1$. The value of α can be reduced to accommodate for a larger error in odometry and distance measurements.

Lastly, the following DVC strategy only requires the use of a desired maximum velocity U (i.e. it does not need a desired maximum acceleration A^+ or deceleration A^- to be set). On the other hand, the parameter tuning guidelines presented require the use of a desired maximum acceleration A^+ or deceleration A^- to calculate $\hat{c}$. This requirement can be bypassed by combining Equations (4.7) and (4.8), such that

$$\hat{c} = \frac{\beta \times U}{\sqrt{d}}. \tag{4.15}$$

Note that, since this approach does not take into account a desired maximum acceleration A^+ or deceleration A^-, it assumes that the maximum acceleration and deceleration of the physical robots are very large and will not affect the control of the robot in any way. If this is not the case, then the parameter tuning guidelines should be instead used as presented.

The DVC strategy follows three main steps:
Step 1: As it was previously discussed, since each accelerating term contains a random parameter $\mathbf{r}$, the obstacle susceptibility term can end up being small while the robot is close to an obstacle. If that happens while the other accelerating terms are large, collisions with obstacles can occur. Therefore, the first step aims to address the case where the robot is not repelled by the closest obstacle at a given timestep, because $\mathbf{r_3}$ happens to be small. To avoid collision, the effect of the cognitive and social terms (first and second accelerating terms) must be small enough to ensure that at least for the next timestep, it will be impossible for the robot to collide with the closest obstacle. In this case, c_3 is ignored (since the obstacle susceptibility term is assumed to be small) and $\hat{c} = c_1 + c_2$.

- Calculate the desired maximum speed $U_{1,2} = \alpha \times \nu_1$ that will ensure no collision with the closest obstacle.
- Using (4.15), calculate $\hat{c}$, where $d = 2$.
- Using $\hat{c} = c_1 + c_2$, calculate c_1 and c_2 based on the desired ratio $\frac{c_1}{c_2}$.

Step 2: Another problematic case can be identified where, while the robot is repelled by the closest obstacle, it ends up colliding with another obstacle. To avoid this, the

effect of all three accelerating terms needs to be small enough to ensure that at least for the next timestep, it will be impossible for the robot to collide with the second closest obstacle. In this case, $\hat{c} = c_1 + c_2 + c_3$. As before,

- Calculate the desired maximum speed $U = \alpha \times \nu_2$ that will ensure no collision with the second closest obstacle due to the effect of all accelerating terms.
- Using (4.15), calculate $\hat{c}$, where $d = 2$.
- Finally, since c_1 and c_2 are already known, $c_3 = \hat{c} - c_1 - c_2$.

Step 3: The inertia weight ω can be calculated using (4.9).

One important characteristic of this calibration strategy is that when $s_1 = s_2$, then $c_3 = 0$. This means that when the robot is at an equal distance from two obstacles, there is no repulsive effect on the robot, allowing it to pass through the obstacles. This will happen no matter how big the opening is between the obstacles, as long as the robot can fit through it.

4.3 Simulations

To demonstrate the performance of Adapted RPSO and the DVC strategy proposed in Section 4.2.2, a number of simulations were performed in MATLAB and Gazebo (Koenig & Howard 2004). The MATLAB simulations were idealised, whereas the Gazebo simulations included a more detailed real-time physics model where the inertia of the robots is applied.

Firstly, the tested RPSO variants will be described. The cognitive coefficient c_1 allows the robot to explore the environment around it and overcome obstacles, while the social coefficient c_2 encourages the robot to move towards the global best location $\mathbf{y_g}$. As both coefficients are of importance in source localisation tasks it is assumed in all of the following simulations that

$$c_1 = c_2. \tag{4.16}$$

This provides the ratio $\frac{c_1}{c_2}$ required in Step 1 of the DVC strategy. The values for c_1, c_2 and c_3 can be either constant (calibrated at the beginning of the simulation) or dynamic (i.e. re-calibrated at every timestep using the DVC strategy). Both scenarios will be studied in the following simulations. The algorithms compared in the simulations are described below.

Original RPSO with constant values For the original RPSO algorithm, the magnitudes of the single values c_1, c_2 and c_3 rarely matter. This is because it is very easy for the resulting velocity of the algorithm to be higher than the maximum velocity of the robot (since the accelerating terms are unbounded). Instead, what matters is the ratio $\frac{c_3}{c_1+c_2}$, since this will control the direction of the requested velocity, which will be either away from close obstacles or towards the personal and global best locations. For the following simulations, three cases will be tested: $c_3 \approx c_1 + c_2$, $c_3 \gg c_1 + c_2$, and $c_3 \ll c_1 + c_2$.

Adapted RPSO with constant values In Adapted RPSO, all of the terms of the velocity update equation are bounded using the sgn function. Therefore, it is possible to tune the parameters using Equation (4.9). The parameters are tuned so that the desired maximum velocity U is always equal to the physical maximum velocity of the robots. From here, it is now possible to calculate the values of c_1, c_2 and c_3, based on the desired value of the ratio $\frac{c_3}{c_1+c_2}$. In order to allow direct comparison between the algorithms, the same cases will be used for the Adapted RPSO with constant values, as for the original RPSO, where it will be ensured that different values of $\frac{c_3}{c_1+c_2}$ are tested.

Adapted RPSO with DVC For Adapted RPSO with DVC, c_1, c_2 and c_3 will be recalibrated at every timestep for each robot, depending on its current state as explained in Section 4.2.2. By running Adapted RPSO twice, once with constant parameters and once with DVC, it will be possible to understand the effect that DVC has on the performance of the algorithm.

The Adapted PSO controller with the DVC strategy used in the simulations is shown in Algorithm 1, where f refers to the fitness of the personal best location of a robot and f_g refers to the fitness of the global best location. For Adapted PSO with constant parameters, the same controller was used but without recalibration of the PSO parameters (lines 12-18 in Algorithm 1), while for original PSO with constant parameters, the PSO parameters are not recalibrated and the PSO velocity updated equation (line 19) is replaced by the original PSO equation.

The simulations were created to resemble typical real-world robotic applications (e.g. a swarm of drones navigating through a forest or a city). The values of some of the parameters used (α, β, γ, Δt and ω) were selected heuristically to match such applications. Changing the value of the parameters β, γ, Δt and ω is expected to affect all cases in the same way. In the case of α, it is only used by one of the tested cases (Adapted RPSO with DVC) and it is used to accomodate for errors in odometry and

Algorithm 1: Robot Control using Adapted PSO with DVC

foreach *robot* **do**
  robot.UpdatePersonalBestLocation(source_position);
  if *robot.f* $< f_g$ **then**
    $f_g \leftarrow$ robot.f; `// Update global best location fitness`
    $\mathbf{y_g} \leftarrow$ robot.$\mathbf{y}$; `// Update global best location`
  end
end
foreach *robot* **do**
  $s[] \leftarrow$ robot.GetDistanceToSurroundings(6); `// 6 sensing regions`
  $\mathbf{F}[] \leftarrow$ robot.GetForcesFromSurroundings($s[]$);
  $\mathbf{F_t} \leftarrow sum(\mathbf{F}[])$;
  $s[]$.sort();
  $\nu[] \leftarrow s[]/\Delta t$;
  $U_{1,2} \leftarrow \alpha \times \nu[1]$; $U \leftarrow \alpha \times \nu[2]$;
  $\hat{c} \leftarrow \beta * U_{1,2}/\sqrt{2}$; `// First Step:` $\hat{c} = c_1 + c_2$
  $c_1 \leftarrow \hat{c}/2$; $c_2 \leftarrow \hat{c}/2$; `// In this case` $c_1/c_2 = 1$
  $\hat{c} \leftarrow \beta * U/\sqrt{2})$; `// Second Step:` $\hat{c} = c_1 + c_2 + c_3$
  $c_3 \leftarrow \hat{c} - c_1 - c_2$;
  $\omega \leftarrow 1 - \beta$; `// Third Step:` $\hat{c} = c_1 + c_2 + c_3$
  robot.$\mathbf{v} \leftarrow \omega *$ robot.$\mathbf{v} + c_1 * rand(1,2) *$ sgn(robot.$\mathbf{y}$ − robot.$\mathbf{x}$) $+ c_2 *$ $rand(1,2) *$ sgn(robot.$\mathbf{y_g}$ − robot.$\mathbf{x}$) $+ c_3 * rand(1,2) *$ sgn($\mathbf{F_t}$);
end

Function *UpdatePersonalBestLocation(self, source_position)* **:**
  self.$\mathbf{x} \leftarrow$ self.GetCurrentPosition();
  $f =$ |self.$\mathbf{x}$−source_position|; `// Assign distance-based cost to location`
  if $f <$ *self.f* **then** `// Update personal best location`
    self.$f \leftarrow f$;
    self.$\mathbf{y} \leftarrow$ self.$\mathbf{x}$;
  end
end

Table 4.1: Table of values used for different parameters

Algorithm	Case	α	β	γ	Δt	$c_1 = c_2$	$c3$
Adapted RPSO	DVC	0.9	0.9	1	1	-	-
	$c_3 \gg c_1 + c_2$	-	0.9	1	1	0.2864	1.1455
	$c_3 \approx c_1 + c_2$					0.4296	0.8591
	$c_3 \ll c_1 + c_2$					0.5728	0.5728
Original RPSO	$c_3 \gg c_1 + c_2$	-	0.9	1	1	0.2864	1.1455
	$c_3 \approx c_1 + c_2$					0.4296	0.8591
	$c_3 \ll c_1 + c_2$					0.5728	0.5728

distance measurements. In the following simulations it is assumed that such errors are small and therefore a large value of α is used. Further tests with sensors of larger error would be useful but are beyond the scope of this thesis. Table 4.1 shows the values assigned to these parameters throughout all of the simulations, along with the values of the parameters c_1, c_2 and c_3 for each tested case.

4.3.1 World description

The world used in all simulations is shown in Figure 4-1. In the figure, the blue square is the area where the robots are initialised, the red square is the source (global minimum of the cost function, where the cost is equal to the distance between the robot and the source) and the circles are obstacles. The obstacles become denser the closer to the source.

The robots can move within the two-dimensional world, are assumed to have unlimited communication range and bandwidth, and each robot can communicate with every other robot in the swarm at all times. To assess the performance of each swarm, the fitness of the swarm is calculated using

$$fitness = \frac{\sum_{i=1}^{M} x_1^i}{M}, \tag{4.17}$$

where M is the total number of robots in the swarm and x_1^i is the horizontal component of the position of robot i. Therefore, the right-hand side of the equation is the horizontal distance from the origin to the Centre of Mass (CoM) of the swarm. Therefore, the further a swarm manages to navigate inside the obstacle course (higher fitness), the more capable the control algorithm is to deal with smaller openings. Furthermore, collisions with obstacles or other robots can occur and if a collision occurs the robot is considered to become disabled and cannot move any further. Apart from the fitness of the swarm, the percentage of robots that have collided by the end of the simulation is

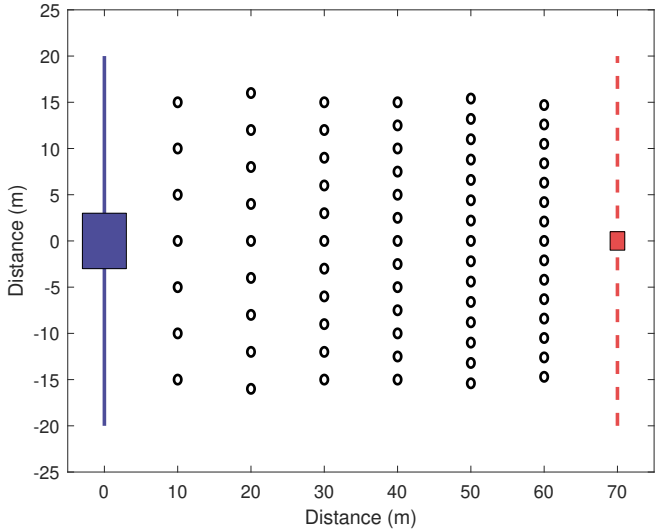


Figure 4-1: The obstacle map used in both MATLAB and Gazebo simulations. The blue square on the left shows the starting area where robots are initialised and the red square on the right shows the position of the source. The obstacles become denser the closer to the source.

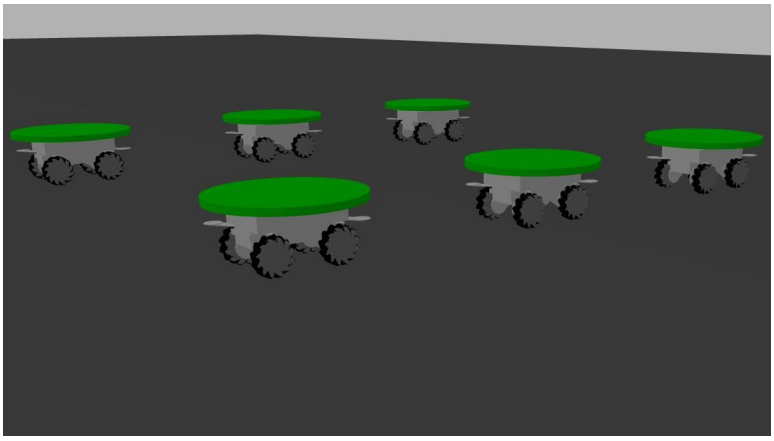
also used as a secondary metric.

4.3.2 Robot description

Although all tested algorithms are inherently scalable, a minimum number of robots is required for the swarm to be effective. In this simulations, it was chosen to demonstrate the algorithms on a small swarm of 6 robots. The robots of the swarm are based on the Robotnik Summit XL Steel platform (Robotnik Automation S.L.L. n.d.), a popular robotic platform shown in Figure 4-2 with available specifications and simulation models (e.g. Gazebo models (The Construct 2021)).

The robots detect nearby obstacles using a LiDAR sensor. A low-cost obstacle-detection short-range LiDAR sensor can have maximum range starting from 4 m (Benewake (Beijing) Co. Ltd 2017), so the obstacle detection range of each robot was set to 3 m, to avoid operating at the sensor's maximum range. Due to the way that both the MATLAB and Gazebo simulations operate, a robot can detect obstacles in its detection range even if they are "hidden" behind other obstacles (in the Gazebo simulations, the robots are already aware of the location of the obstacles). This contradicts how a LiDAR would detect obstacles and it can result in multiple repulsive forces being exerted from the same direction. To avoid this, each robot separates its surroundings into six equally-sized radially-separated regions, as shown in Figure 4-3. Only the repulsive

(a)



(b)

Figure 4-2: (a) shows the real Robotnik Summit XL Steel platform (Robotnik Automation S.L.L. n.d.), while (b) shows a simulated modified model of the platform used in some of the following simulations. The robots are equipped with mecanum wheels for holonomic motion and a contact sensor (green link) to detect collisions.

forces exerted by the closest body in each region are accounted for the calculation of the total repulsive force. The sensing regions along with the operation of the proposed DVC strategy are illustrated in Figure 4-3.

The tested Adapted RPSO algorithms needed to be properly tuned to match the physical limitations of the robots. Therefore, the maximum desired velocity U is limited to 3 m/s (the actual maximum speed of the Summit XL Steel). The sensitivity parameter β is set to 0.9 (as shown in Chapter 3, a large β can allow more effective control of the robot). The actual maximum acceleration of Summit XL Steel is not available but it is assumed to be high, since it uses electric motors that are characterised by high acceleration and therefore it is assumed that it does not interfere with the control of the robots in any significant way (i.e. the motion of the robot can be adequately modelled using a kinematic motion model instead of requiring a dynamic one). Since the maximum acceleration can be ignored, the accelerating coefficients are tuned using only U, as in Equation (4.15).

4.3.3 Gazebo Simulations

The MATLAB simulations were inherently simplistic, to allow for a large number of simulations to be run, in order to obtain strong statistical significance of the behaviour of each algorithm. The position and velocity of the robots are calculated in every timestep using the velocity and position update Equations (4.5) and (4.6) respectively. Therefore, the simulations assume that the robots behave like perfect PSO particles. To validate these simplistic simulations, a number of more realistic simulations were run in Gazebo (benefiting from a detailed Physics Engine). An instance of such a simulation is shown in Figure 4-4. In these simulations, the RPSO controller outputs a specific velocity demand, calculated using Equation (4.5) and the robot actuates itself to meet this demand. The robots move by forces being applied on them and they have inertia.

In Gazebo, the robots are equipped with contact sensors to detect collisions (green links shown in Figure 4-2b). Furthermore, since the demanded velocity can have any direction, the robots are equipped with mecanum wheels to allow for holonomic motion. The motion of the robots is governed by the forward and inverse kinematic equations of the Mecanum wheels (Taheri et al. 2015)

$$\begin{bmatrix} v_x \\ v_y \\ w_z \end{bmatrix} = \frac{r}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & -1 \\ \frac{-1}{(l_x+l_y)} & \frac{1}{(l_x+l_y)} & \frac{-1}{(l_x+l_y)} & \frac{1}{(l_x+l_y)} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} \tag{4.18}$$

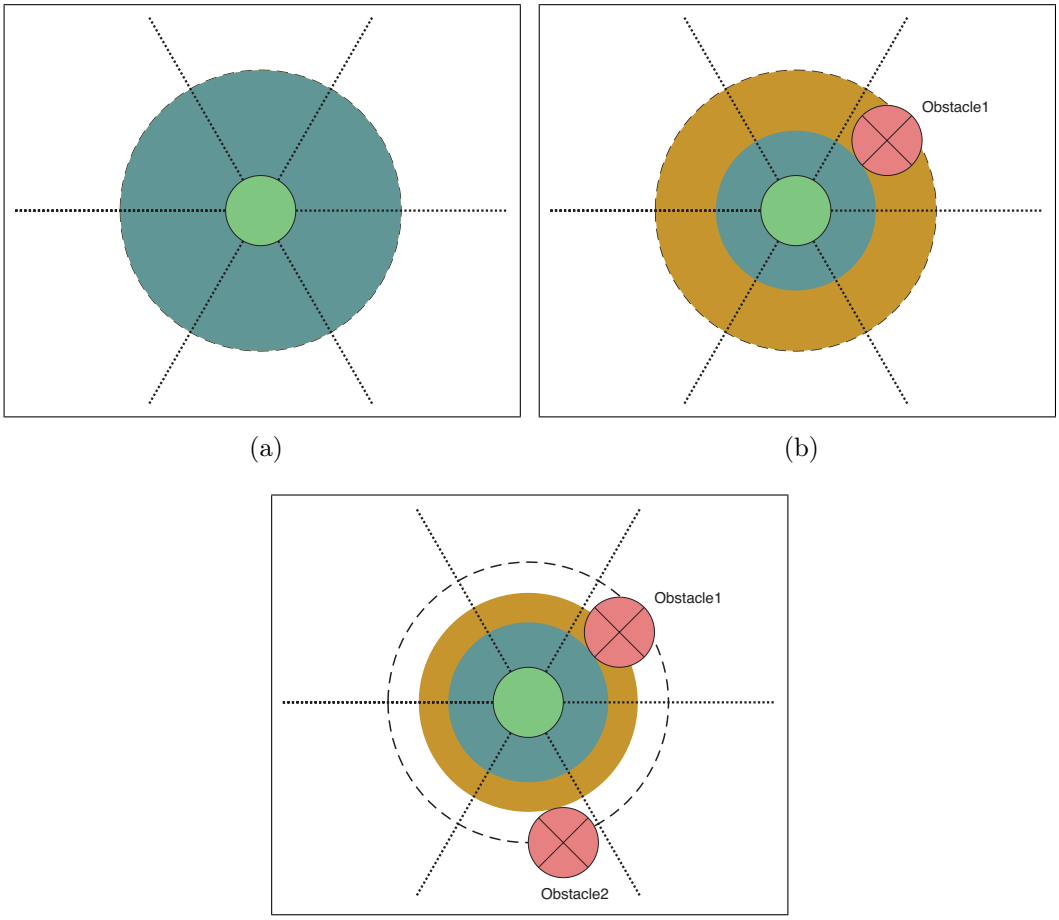


Figure 4-3: The three possible cases that are covered by the calibration strategy during DVC. The green circle represents the robot and the red crossed circles are the obstacles. The dotted lines show how the six sensing regions are separated and the dashed circle represents the maximum detection range of the robot. If no obstacle is present, as in (a), $c_1 + c_2$ (blue region) is maximised and $c_3 = 0$. If one obstacle is present, as in (b), $c_1 + c_2$ is limited by the distance to the obstacle; c_3 (orange region) is increased to cover the extra potential for movement. If two or more obstacles are present, as in (c), $c_1 + c_2$ is limited by the distance to the closest obstacle and c_3 (orange region) is increased to cover the extra potential for movement but it is also limited by the distance to the second closest obstacle.

and

$$\begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} = \frac{1}{r} \begin{bmatrix} 1 & -1 & -(l_x + l_y) \\ 1 & 1 & (l_x + l_y) \\ 1 & 1 & -(l_x + l_y) \\ 1 & -1 & (l_x + l_y) \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ w_z \end{bmatrix}, \tag{4.19}$$

where v_x and v_y are the forward/backward and lateral velocities of the robot, w_z is its angular velocity, w_1, w_2, w_3 and w_4 are the angular velocities of each wheel and r is the radius of the wheels. The distance l_x is half the distance between the two front wheels and l_y is half the distance between the front wheel and the rear wheel of a given side, such that $\sqrt{l_x^2 + l_x^2}$ is the distance of each wheel from the centre of the robot.

It should be noted that Mecanum wheels do not offer perfect holonomic motion. The maximum possible velocity that the robot can achieve depends on its direction of motion, maximised when the robot moves forwards/backwards or laterally. As the direction of motion becomes more diagonal (i.e. $45^o, 135^o, -135^o$ or -45^o relative to the orientation of the robot), the maximum possible velocity decreases. Therefore, for some directions of motion, the robots may not be capable of meeting the RPSO controller's velocity demands. When such a case is detected, the robot instead moves with the largest velocity that it can achieve. This limitation of the maximum speed of the Gazebo robots can result in slower convergence of the Gazebo swarms compared to the MATLAB ones.

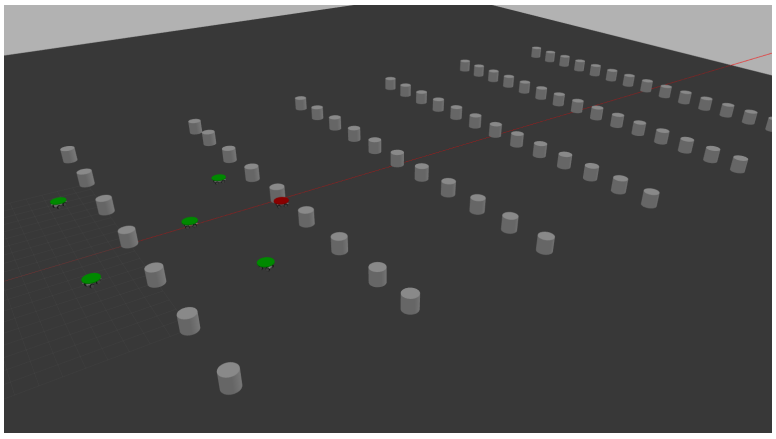


Figure 4-4: Image of the Gazebo environment during the operation of the obstacle course simulations. The global minimum of the fitness function (source) is assumed to be located on the right side of the image outside the obstacle course (the source is not represented by an actual object in order to avoid collisions with the robots). The grey cylinders are the obstacles while the green and red circles are the robots. The green robots are currently operational while the red robot has collided with an obstacle.

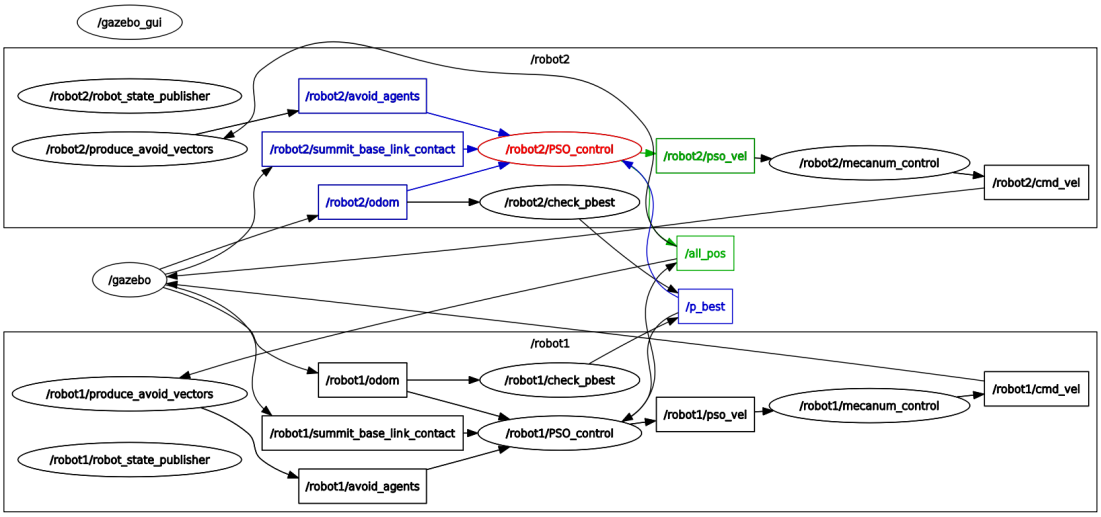


Figure 4-5: Graph of the flow of information between ROS nodes (circles) and ROS topics (rectangles) for a swarm of 2 robots, during the obstacle course simulations. For the operation of the PSO algorithms, each robot maintains four nodes (PSO_control, check_pbest, mecanum_control and produce_avoid_vectors) and four local topics (odom, avoid_agents, pso_vel and cmd_vel). There also exist two global topics that can be accessed by all robots (all_pos which contains the current positions of all robots and p_best which contains the current personal best locations of all robots).

In Gazebo, the robot controller is implemented using the Robot Operating System (ROS) framework. Figure 4-5 shows the nodes and topics used by ROS. For simplicity only the nodes and topics of two robots are included in the figure. The *PSO_control* node is the main node where the RPSO controller is implemented and all decisions are made. The *check_pbest* node is the node responsible for assigning a cost to the current location of the robot and the *produce_avoid_vectors* node calculates the virtual forces $\mathbf{F}$ exerted on the robot by nearby bodies. Lastly, the *mecanum_control* node converts the velocity demand outputted by the RPSO controller into the corresponding values of v_x and v_y and then into the power output for each wheel, using Equation (4.19). All nodes responsible for the operation of a robot are local to the robot and therefore, all decision making, collection of data and actuation happens on the individual level.

Most topics used by ROS are local to each robot, used for the communication of the nodes of the robot. For inter-robot communications, two global topics are used, the *all_pos* topic and the *p_best* topic. The former is responsible for storing the current locations of all robots, while the latter is responsible for storing the current personal best locations of all robots.

One advantage that is offered by ROS is that it can easily allow the control of the

timestep size Δt. This is achieved through the use of the ROS Rate command with input $\frac{1}{\Delta t}$. Since the RPSO controller is implemented in the *PSO_control* node, the refresh frequency of this node is controlled in this way. The rest of the nodes operate at a refresh frequency ten times larger than *PSO_control*, to ensure that all information required by the main node are up-to-date at all times.

4.4 Results

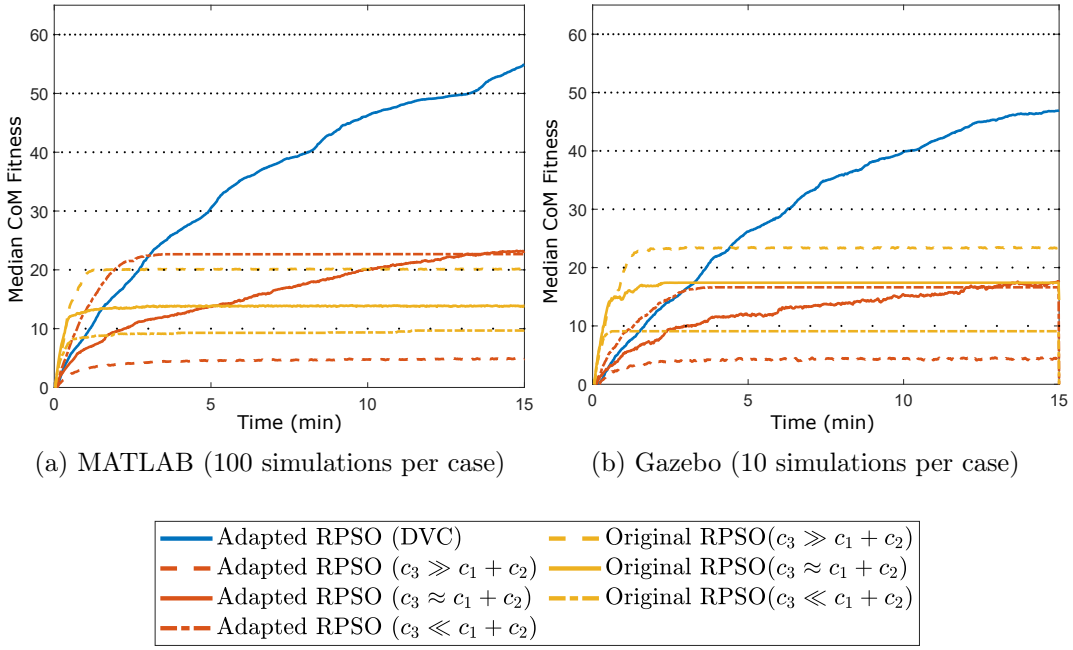
The previous section described in detail both the MATLAB and Gazebo simulations used. This section will present the results from both of these simulations. First, the results from the MATLAB simulations will be presented fully and discussed to draw conclusions about the behaviour of each individual algorithm. Then the Gazebo results will be presented for comparison.

MATLAB was used to simulate the performance of the swarm over 100 repeats, to obtain clear behavioural patterns for each algorithm. Figure 4-6a shows the median CoM fitness over time for each algorithm and Figure 4-7a shows the median number of collided vs operational robots at the end of the simulation. As it can be seen from the results of Figure 4-6a, with Adapted RPSO with DVC, the CoM of the average swarm manages to pass through the fifth layer of obstacles (fifth dotted line) before the end of the simulation. This is in contrast to the other algorithms that do not manage to pass through the third layer. All cases of the original RPSO appear to progress quickly at the beginning. This is in fitting with the fact that the original RPSO almost always operates at the maximum velocity permitted by the physical constraints of the robot. On the other hand, all cases of the Adapted RPSO (including DVC) progress more slowly.

In Figure 4-7a, it can be seen that all cases of both the original RPSO and the Adapted RPSO follow the predicted behaviour. That is, as c_3 gets larger compared to c_1+c_2, the number of collisions decreases. For small c_3, Adapted RPSO results in only collisions, while for large c_3, it results in no collisions. All the cases of original RPSO however have very low survivability.

Lastly, the only cases that end up with absolutely no collisions are the Adapted RPSO with large c_3 and the Adapted RPSO with DVC. Comparing the two cases in Figure 4-6a, it can be seen that the Adapted RPSO with large c_3 has the lowest overall fitness out of all cases. In contrast, the Adapted RPSO with DVC has the highest overall fitness. This shows that the Adapted RPSO with DVC completely overshadows all other cases, both in terms of fitness and robot survivability. This is attributed to the DVC strategy

used. The strategy makes use of the velocity boundaries introduced by Adapted RPSO, to slow down a robot in the presence of an obstacle, making it unlikely to collide with any obstacles or other agents. At the same time, the robot is still capable of navigating through small openings; a capability that is not shared by the other two algorithms.

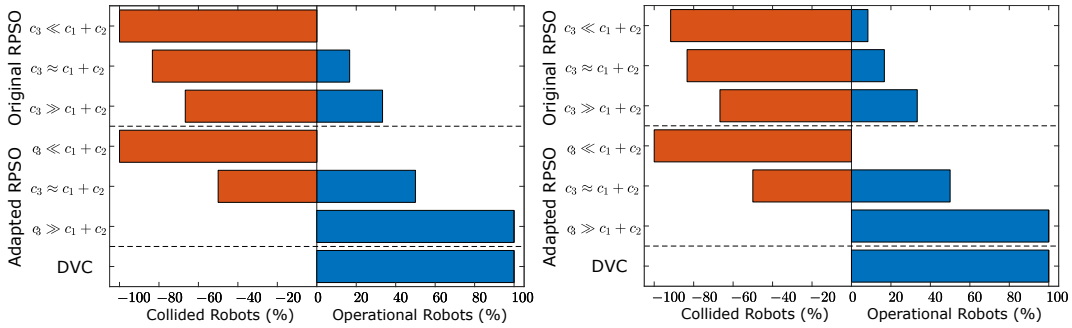


(a) MATLAB (100 simulations per case)

(b) Gazebo (10 simulations per case)

Figure 4-6: Median CoM fitness over time results for different cases. The dotted lines represent the obstacle layers of the obstacle course.

Figures 4-6b and 4-7b show the median CoM fitness over time and the median number of collided vs operational robots for each tested case respectively, for the Gazebo simulations. Comparing Figures 4-6a and 4-6b, it can be seen that there are small differences. Namely, there is a small reduction in the overall performance of the Adapted RPSO cases, which can be probably attributed to the imperfect motion of mecanum wheels (i.e. the maximum speed of the robot is limited when it is moving towards certain directions). However, the Adapted RPSO with DVC performs better than the other algorithms, reaching an average fitness of 47 by the end of the simulation. When it comes to Figures 4-7a and 4-7b, the results look almost identical for all cases. The original RPSO with $c_3 \ll c_1 + c_2$ appears to have limited survivability that is not observed in the MATLAB simulations.



(a) MATLAB (100 simulations per case) (b) Gazebo (10 simulations per case)

Figure 4-7: The expected number of collided vs operational robots at the end of the median simulation for different cases.

4.5 Discussion and Comparison with other SIA

The results of Section 4.4 show that Adapted RPSO offers more reliable and predictable control over the original RPSO. This can be seen from the results of Figure 4-7a where changing the ratio $\frac{c_3}{c_1+c_2}$ results in large predictable changes in the behaviour of Adapted RPSO. By using $c_3 \ll c_1 + c_2$, Adapted RPSO results in full collisions, while for $c_3 \gg c_1 + c_2$, it results in no collisions. On the other hand, for the same values, the original RPSO algorithm exhibits smaller differences in its behaviour, which suggests that it is more difficult for this algorithm to control effectively the behaviour of the robots.

That said, the main advantage of Adapted RPSO is its ability to use a DVC strategy. This allows the control of the individual terms of Adapted RPSO, resulting in more flexible motion control. Therefore, it can be concluded that, like Adapted RPSO, any version of Generalised Adapted PSO will be more capable of robotic motion control than any equivalent version of the original PSO algorithm (i.e. for any number of accelerating terms).

Apart from the original PSO, it is also desirable to compare Generalised Adapted PSO to all other SIA discussed in Chapter 2. Qualitative comparisons will be offered in the rest of this section.

Glowworm Swarm Optimisation: Apart from PSO, GSO is the SIA that was mostly studied for use in robotic swarms, as discussed in Chapter 2. This is because GSO was designed to address some of the main problems of PSO, like its lack of a maximum velocity limitation. As a reminder, the motion of a glowworm in GSO is

described by the following position update rule

$$\mathbf{x}^i[k+1] = \mathbf{x}^i[k] + s\frac{\mathbf{x}^j[k] - \mathbf{x}^i[k]}{\|\mathbf{x}^j[k] - \mathbf{x}^i[k]\|}, \tag{4.20}$$

where s is the step size, which controls the maximum change in position that can occur for each glowworm. Therefore, provided that the GSO timestep size Δt is constant, s can be used to control the maximum velocity of the glowworm. That said, the way that GSO addresses this problem, introduces other weaknesses into the algorithm. GSO lacks a stochastic component in its position update equation, which is an important part of swarm optimisation algorithms to allow the individual to escape local minima. On the other hand, Generalised Adapted PSO maintains stochastic components in all of its accelerating terms.

Additionally, the parameter s is the only parameter in GSO and it controls both the maximum velocity and maximum acceleration of the glowworm. In this way, a single value of s corresponds to only a specific pair of maximum velocity and maximum acceleration (their relationship depends on the timestep size Δt), which reduces the flexibility of the algorithm and prevents it from being properly synchronised with different robotic platforms.

Firefly Algorithm: FA was introduced to address the lack of stochastic components in GSO. The motion of a firefly in FA is described by

$$\mathbf{x}^i[k+1] = \mathbf{x}^i[k] + \beta_0 e^{-\gamma r_{ij}^2}(\mathbf{x}_j[k] - \mathbf{x}_i[k]) + \alpha\boldsymbol{\epsilon}^i[k], \tag{4.21}$$

The vector $\boldsymbol{\epsilon}$ is a random vector while the parameter α is a tunable parameter that controls the magnitude of $\boldsymbol{\epsilon}$. The term $\alpha\boldsymbol{\epsilon}^i[k]$ is therefore a stochastic term added to the position update equation, while other terms are fully deterministic.

As discussed in Chapter 2, the stochastic component of FA can affect the quality of control that the algorithm can offer. This is because the stochastic term has random direction, which can either accelerate or decelerate the firefly towards any given direction. Therefore, the maximum velocity and acceleration of the algorithm may be exceeded thereby resulting in unpredictable behaviour and collisions. On the other hand, each stochastic component in Generalised Adapted PSO can only reduce the accelerating effect of its corresponding term, thereby maintaining the maximum velocity and acceleration limitations as defined by U and A^+.

A-CMOMMT: This SIA was designed to consider virtual forces exerted by multiple targets and robots of the swarm, by summing them up using the following equation

$$\sum_{t=1}^{T} w_{i,t}\mathbf{f}_{i,t} + \sum_{j=1}^{J} \mathbf{g}_{i,j}, \tag{4.22}$$

where T is the number of targets around robot i, J is the number of other robots around robot i and $\mathbf{f}_{i,t}$ and $\mathbf{g}_{i,j}$ are the functions that calculate the virtual forces. As discussed in Chapter 2, the problems of A-CMOMMT are that, like GSO, it does not include any stochastic components and that it is important to properly tune $\mathbf{f}_{i,t}$ and $\mathbf{g}_{i,j}$, so that one does not overshadow the other.

This chapter has shown how Generalised Adapted PSO can be used to avoid collisions with multiple obstacles and other robots using virtual forces, through the use of an additional accelerating term. An equivalent way could be used to sum up forces exerted by multiple targets. Alternatively, Generalised Adapted PSO can take a similar form to A-CMOMMT by dedicating one accelerating term to each virtual force (either exerted by a target or another robot), with the difference that each term will also include a stochastic component.

When it comes to the merging of the forces, in Generalised Adapted PSO each force is passed through the sgn function before any other operation is carried out on it. This normalises the magnitude of each virtual force, while maintaining its direction. Therefore, there is no risk that one type of virtual force will overshadow another. Instead, the effect of each individual force will be controlled by a properly designed DVC strategy, similar to the one presented in this chapter.

This last concept can be extended beyond the use of simple virtual forces, to any function that outputs a direction of motion. In a similar manner to how obstacle avoidance was implemented in this chapter, other tasks can be also incorporated as additional accelerating terms. In this way, Generalised Adapted PSO can take the form of a task managing framework, where off-the-shelf algorithms responsible for different swarm robotic tasks can be merged together, without the need for any additional tuning and modifications. This can allow the inclusion of a large number of tasks into the swarm control algorithm; a capability that has not been possible so far in swarm robotics, as discussed in Chapter 2.

4.6 Conclusions

This chapter has described how the Generalised Adapted PSO algorithm can be used to incorporate aggregation and obstacle avoidance into PSO through the use of an additional accelerating term. The results of the presented simulations show that Generalised Adapted PSO offers more reliable and predictable control than original PSO. Furthermore, the capability of Generalised Adapted PSO to make use of a DVC strategy, greatly increases the flexibility of the algorithm.

Section 4.5 presented a comparison of Generalised Adapted PSO with other SIA. Overall, Generalised Adapted PSO shares all of the discussed advantages of other SIA, while also addressing all of their disadvantages. In the end, it is also discussed how Generalised Adapted PSO can be used to incorporate other swarm robotic tasks, beyond obstacle avoidance and aggregation into PSO, thereby addressing a major gap in the swarm robotic literature, the appropriate merging of a large number of different tasks.

References

Benewake (Beijing) Co. Ltd (2017), 'CE30 3D obstacle-avoidance LiDAR'.
URL: *http://en.benewake.com/product/detail/5c34571eadd0b639f4340ce5.html*

Couceiro, M. S., Rocha, R. P. & Ferreira, N. M. F. (2011), A novel multi-robot exploration approach based on Particle Swarm Optimization algorithms, *in* '2011 IEEE International Symposium on Safety, Security, and Rescue Robotics', pp. 327–332.

Garone, E., Nicotra, M. & Ntogramatzidis, L. (2018), 'Explicit reference governor for linear systems', *International Journal of Control* **91**(6), 1415–1430.
URL: *https://doi.org/10.1080/00207179.2017.1317832*

Hosseinzadeh, M. & Garone, E. (2020), 'An Explicit Reference Governor for the Intersection of Concave Constraints', *IEEE Transactions on Automatic Control* **65**(1), 1–11.

Khaldi, B. & Cherif, F. (2016), A virtual viscoelastic based aggregation model for self-organization of swarm robots system, *in* 'Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)', Vol. 9716, pp. 202–213.

Koenig, N. & Howard, A. (2004), Design and use paradigms for Gazebo, an open-source multi-robot simulator, *in* '2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)', Vol. 3, pp. 2149–2154 vol.3.

Robotnik Automation S.L.L. (n.d.), 'SUMMIT-XL STEEL MOBILE ROBOT'.
URL: *https://robotnik.eu/products/mobile-robots/summit-xl-steel-en/*

Spears, W. M., Spears, D. F., Hamann, J. C. & Heil, R. (2004), 'Distributed, Physics-Based Control of Swarms of Vehicles', *Autonomous Robots* **17**(2), 137–162.
URL: *https://doi.org/10.1023/B:AURO.0000033970.96785.f2*

Taheri, H., Qiao, B. & Ghaeminezhad, N. (2015), 'Kinematic model of a four mecanum wheeled mobile robot', *International journal of computer applications* **113**(3), 6–9.

The Construct (2021), 'Exploring the World with Mecanum Wheels: Introduction to Mecanum Wheels'.
URL: *https://app.theconstructsim.com/#/l/4236960c/*

Chapter 5

Robot-Centred Reference Frame and the Non-Omnidirectional PSO Controller

The previous chapters have shown how a robotic swarm can be controlled by merging different tasks using Generalised Adapted PSO and assigning priority to them using a properly designed DVC strategy, while considering the physical limitations of the robots. That said, one limitation can be seen from the way Generalised Adapted PSO is used in Chapter 4. One main characteristic that is inherited from the original PSO algorithm is that velocity outputs can have any arbitrary direction. Therefore, to achieve optimal motion control, the robots used need to be capable of holonomic motion (i.e. fully-actuated robots such as robots that use Mecanum wheels like in the simulations of Chapter 4).

This approach can be limiting, considering the large number of non-holonomic robots (i.e. underactuated robots). Especially in the case of marine robotics, robotic platforms tend to be separated into two main categories, 1) Autonomous Underwater/Surface Vehicles (AUV/ASV) that are generally faster but non-holonomic (underactuated) and 2) Remotely Operated Vehicles (ROV) that can be holonomic (fully-actuated) but are usually much slower (Nad et al. 2015, Njaka et al. 2020). Due to these characteristics, the former category is typically used in applications where large areas need to be searched (Robbins et al. 2006), including applications of target detection (Figueiredo et al. 2014), while the latter is usually restricted for use in small areas or confined spaces like harbours (Choi et al. 2017) or for localised tasks like underwater infras-

tructure inspections (Macreadie et al. 2018, Batlle et al. 2003). Moreover, due to the differences in capabilities of the two classes, research organisations are looking for ways to implement hybrid control strategies where both classes can be used simultaneously (Ludvigsen et al. 2013). Since this thesis is primarily concerned with underwater source localisation over large areas, it is desirable to show how Generalised Adapted PSO can be used in non-holonomic robotic platforms such as AUVs and ASVs. This chapter will aim to address this by adapting the algorithm for use in non-holonomic robots with limited degrees of freedom (e.g. differential drive (Rubio et al. 2019, Bräunl 2008)). To achieve this, the chapter will delve deeper into the particular characteristics of PSO, recognising some inherent problems of the algorithm and it will show how they can be addressed.

5.1 Robot-Centred Reference Frame

As explained in Chapter 3, PSO is an algorithm created for use in numerical optimisation tasks and therefore it has several characteristics specifically selected to simplify its operation by reducing the number and complexity of computational operations. One of these characteristics is its use of a global reference frame (inertial frame) for velocity description and updating. In other words, each particle's velocity components are described with respect to global coordinate axes. Furthermore, each velocity component is decoupled from the others and therefore it is not affected by changes to them. This lack of motion constraints allow for the updating of the velocity components to happen without taking the orientation of the robot into account. As a result, the movement of all particles in the swarm is expressed using only the two relatively simple position and velocity update equations, which are presented below as a reminder for robot i at timestep k,

$$\mathbf{x}^i[k+1] = \mathbf{x}^i[k] + \mathbf{u}^i[k+1], \tag{5.1}$$

$$\mathbf{u}^i[k+1] = \omega \mathbf{u}^i[k] + c_1 \mathbf{r_1} \circ (\mathbf{y}^i[k] - \mathbf{x}^i[k]) + c_2 \mathbf{r_2} \circ (\mathbf{y_g}[k] - \mathbf{x}^i[k]). \tag{5.2}$$

In a simulation environment this can end up saving considerable operational time. On the other hand, the use of a global reference frame along with the decoupling of the velocity components can introduce disadvantages to the algorithm. One trend that is well known is that the particles tend to move along the global coordinate axes (Janson & Middendorf 2007, Spears et al. 2010). This effect is visualised in Figure 5-1, where the simulated particles of PSO can be seen to form cross-shaped structures around the location of the global extremum.

When a global frame of reference is used for the control of robotic swarms, it can cause

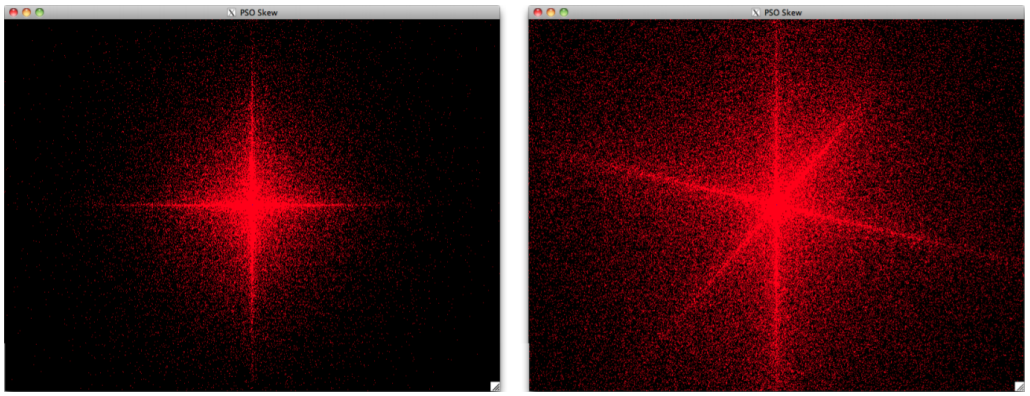


Figure 5-1: Visualisation of the PSO particles for the 2D and the 3D case. The particles tend to gather on the global coordinate axes, forming cross-shaped patterns. This is understood to be the result of the decoupling of the velocity components, along with the use of global coordinate axes. (Spears et al. 2010)

the swarm to move collectively along the direction of one global coordinate axis until it is forced to change direction by an external factor. Furthermore, several additional limitations are introduced. For example, not considering the orientation of the robot when updating its velocity can limit the robot's reactions to its surroundings. This is because the robot needs to treat all obstacles in the same way, no matter if they are located in front of it, behind it or beside it. Moreover, it should be noted that the original advantage of using a global reference frame to reduce computational complexity does not apply to robotic applications. This is because, apart from specific sensor data (e.g. GPS data), most of the spatial data considered by a robot are calculated relative to its current position and orientation (e.g. IMU readings, it's current velocity, position of surrounding objects etc.). Converting these to a global frame of reference can instead introduce additional computational complexity into the robot controller. Considering these limitations can therefore suggest that for the control of robotic swarms, Generalised Adapted PSO should use a robot-centred reference frame (body frame) for the description and updating of the robots' velocity components. The rest of this chapter will show how this can be achieved. Moreover, it will be shown how the use of a robot-centred reference frame, can allow Generalised Adapted PSO to be adapted for the control of non-holonomic robots (e.g. differential drive, traditional steering, forward flight etc.).

5.1.1 Robot-Centred Reference Frame Generalised Adapted PSO

In a robot-centred reference frame, each robot makes use of local coordinate axes, where the origin is located at the current position of the robot and the x-axis is parallel to

the orientation of the robot, as shown in Figure 5-2. Therefore, instead of the global velocity vector $\mathbf{u}$, each robot uses a relative velocity vector $\tilde{\mathbf{u}}$. The first component v_1 of $\tilde{\mathbf{u}}$ represents the robot's forward/backward movement and the other v_2 represents its lateral movement, as shown in Figure 5-2a, such that at timestep k,

$$\tilde{\mathbf{u}}[k] = \begin{bmatrix} v_1[k] \\ v_2[k] \end{bmatrix}.$$

Alternatively, for vehicles that do not need the lateral velocity component v_2 (i.e. robots that employ differential drive, traditional steering etc.), it can be removed. Instead, for such driving mechanisms, it is more useful to consider the angular velocity w of the robot, as shown in Figure 5-2b. To achieve this, the relative velocity vector $\tilde{\mathbf{u}}$ at timestep k is modified to

$$\tilde{\mathbf{u}}[k] = \begin{bmatrix} v[k] \\ w[k] \end{bmatrix},$$

The rest of this chapter will consider this latter definition of $\tilde{\mathbf{u}}$, since it is the most common for non-holonomic robots. Therefore, the Generalised Adapted PSO velocity update equation

$$\begin{aligned} \mathbf{u}[k+1] = \omega\mathbf{u}[k] &+ c_1\mathbf{r_1} \circ \text{sgn}(\mathbf{y_1}[k] - \mathbf{x}[k]) \\ &+ c_2\mathbf{r_2} \circ \text{sgn}(\mathbf{y_2}[k] - \mathbf{x}[k]) \\ &+ ... + c_n\mathbf{r_n} \circ \text{sgn}(\mathbf{y_n}[k] - \mathbf{x}[k]), \end{aligned} \quad (5.3)$$

changes for a robot-centred reference frame into

$$\begin{aligned} \tilde{\mathbf{u}}[k+1] = \boldsymbol{\omega} \circ \tilde{\mathbf{u}}[k] &+ \mathbf{c_1} \circ \mathbf{r_1} \circ \text{sgn}(\tilde{\mathbf{y_1}}[k]) \\ &+ \mathbf{c_2} \circ \mathbf{r_2} \circ \text{sgn}(\tilde{\mathbf{y_2}}[k]) \\ &+ ... + \mathbf{c_n} \circ \mathbf{r_n} \circ \text{sgn}(\tilde{\mathbf{y_n}}[k]), \end{aligned} \quad (5.4)$$

where $\tilde{\mathbf{y_1}}$, $\tilde{\mathbf{y_2}}$, ..., $\tilde{\mathbf{y_n}}$ are locations relative to the current position and orientation of the robot. Note that the inertia weight $\boldsymbol{\omega}$ and the acceleration coefficients $\mathbf{c_1}$, $\mathbf{c_2}$, ..., $\mathbf{c_n}$ are now vectors of length 2 instead of single parameters. The parameter tuning guidelines presented in Chapter 3 can therefore be used to calculate the corresponding components of $\boldsymbol{\omega}$, $\mathbf{c_1}$, $\mathbf{c_2}$, ..., $\mathbf{c_n}$ for the linear and angular velocities independently.

With the velocity update equation defined, it remains to show how the position updating equation changes for the robot-centred reference frame controller. The process of converting the relative velocity $\tilde{\mathbf{u}}$ of the robot into a change to its global position $\mathbf{x}$,

(a) (b)

Figure 5-2: Schematics that show the separation of the velocity components for a robot-centred frame of reference, where the green circle is the robot and the yellow arrow indicates its orientation. In (a), the velocity components v_1 and v_2 represent the longitudinal and lateral linear velocities of the robot respectively, while in (b), the velocity components v and w represent the linear and angular velocities of the robot respectively.

consists of several steps. First, the relative velocity $\tilde{\mathbf{u}}$ is used to calculate the relative displacement $\tilde{\mathbf{\Delta x}}$ of the robot in the next timestep. Then the relative displacement $\tilde{\mathbf{\Delta x}}$ is converted into global displacement $\mathbf{\Delta x}$, which is then used to update the robot's global position $\mathbf{x}$. The process is described in more detail below.

The displacement of the robot at timestep $k+1$ relative to its position and orientation at timestep k is given by

$$\tilde{\mathbf{\Delta x}}[k+1] = r[k] \times \begin{bmatrix} \sin(w[k] \times \Delta t) \\ 1 - \cos(w[k] \times \Delta t) \end{bmatrix}, \tag{5.5}$$

where $r[k]$ is the turning radius at timestep k, given by

$$r[k] = \frac{v[k]}{w[k]}. \tag{5.6}$$

Equation (5.6) is undefined for $w[k] = 0$. Therefore, when this problematic case occurs, (5.5) takes the form

$$\tilde{\mathbf{\Delta x}}[k+1] = \begin{bmatrix} v[k] \times \Delta t \\ 0 \end{bmatrix}, \tag{5.7}$$

thereby representing a forward-backward displacement. The global displacement (i.e

displacement with respect to the global coordinate axes) is therefore calculated by rotating $\tilde{\mathbf{\Delta x}}[k+1]$ by the orientation of the robot,

$$\mathbf{\Delta x}[k+1] = \begin{bmatrix} \cos(o[k]) & \text{-}\sin(o[k]) \\ \sin(o[k]) & \cos(o[k]) \end{bmatrix} \tilde{\mathbf{\Delta x}}[k+1], \tag{5.8}$$

where $o[k]$ is the orientation of the robot at timestep k. Finally, the global position $\mathbf{x}$ is given by

$$\mathbf{x}[k+1] = \mathbf{x}[k] + \mathbf{\Delta x}[k+1] \tag{5.9}$$

Additionally, since the orientation of the robot needs to be tracked when a robot-centred reference frame is used, an orientation update equation is also required which is given by

$$o[k+1] = o[k] + w[k] \times \Delta t \tag{5.10}$$

As it can be seen, the position updating process for the robot-centred reference frame PSO is slightly more complex than the global reference frame PSO used in the previous chapters. Nevertheless, the PSO position update equation is only used in simulations where the position of the robot needs to be re-calculated at every timestep. In the real-world or in simulation environments like Gazebo, this position updating process will not be used as it will be carried out by the robot/environment itself. The resulting robot-centred reference frame Generalised Adapted PSO is now capable of controlling the linear and angular velocities of each robot at each timestep. Like the global reference frame Generalised Adapted PSO, this approach can now be adapted for any use, depending on the different tasks that the swarm will need to carry out. The next section will show how a swarm of non-holonomic robots with limited degrees of freedom (e.g. differential drive) can be controlled, where the tasks of obstacle avoidance and source localisation will be combined together.

5.2 Obstacle Avoidance and Dynamic Velocity Control

Chapter 4 described how the global frame of reference Generalised Adapted PSO can be used to combine obstacle avoidance with source localisation, forming the Adapted RPSO algorithm, which will be referred to as the omnidirectional PSO controller for the rest of this chapter. This section will now show how the same result can be achieved using the robot-centred reference frame version of Generalised Adapted PSO, described in Section 5.1.1.

In the omnidirectional PSO controller used in Chapter 4, each task exerts a virtual force

on the robot, altering its linear velocity and pulling it towards a desired direction or pushing it away from obstacles or other robots. The difference with using Equation (5.4) is that it does not only consider the linear velocity of the robot, it also considers its angular velocity. Therefore, for a virtual force to have an effect on the angular velocity of the robot, it must be converted into virtual torque. To achieve this, a virtual rod is assumed to be extended, from the centre of the robot forwards, and the virtual forces are exerted on the edge of the rod, effectively causing the rotation of the robot. The length of the rod does not matter, as in Equation (5.4) both virtual forces and virtual torques pass through a sgn function, making their magnitudes irrelevant. The effect of each virtual torque will instead be controlled by the DVC strategy.

In theory, a non-holonomic robot could be controlled, using Equation (5.4) and virtual forces, in the same way that the holonomic robots were controlled in Chapter 4. Forces exerted on the front or back of the robot will cause it to move longitudinally, while lateral forces will act as torques, causing it to rotate. That said, this way of controlling the robot would share some of the limitations of the omnidirectional PSO controller.

In Chapter 2, it was explained how the dPSO algorithm manages to allow single robot exploration, a capability that is not shared by the original algorithm. In dPSO this result is achieved because the robots cannot always follow the PSO controller's demanded direction of motion and they need to execute circle manoeuvres to align themselves properly, causing them to explore the environment further. This beneficial behaviour occurs beyond the control of the PSO controller and therefore it cannot be predicted or controlled properly. It is therefore desirable to try to emulate this behaviour in such a way that the PSO controller remains in full control of the robot's motion.

To achieve this, this chapter will present a slightly different control approach to the one presented in Chapter 4. Here, a non-omnidirectional PSO controller will be introduced that will aim to maximise the robot's linear velocity v (i.e. continuous forward thrust). The linear velocity v will be limited only in the presence of obstacles. In this way, the robot is prevented from settling down at a single location and is forced to constantly keep moving at maximum speed, thereby causing exploration of the surrounding environment.

First will be described how the non-holonomic robot's linear and angular velocities can vary. The following motion model makes several assumptions to approximate the behaviour of a non-holonomic robot that is only capable of forward motion, rotation around its vertical axis, or a combination of the two. This model will then be used to assemble an appropriate PSO velocity update equation and DVC strategy. These

assumptions are important because, as previously discussed, the linear and angular velocity outputs of Equation (5.4) are decoupled from each other. Therefore, there is always a chance that the PSO controller will request large linear and angular velocities that when combined may exceed the maximum power output of the robot's actuators. The presented motion model will offer a way to detect when this happens through the use of kinetic energies and address it by reducing the linear and angular velocities demanded. The assumptions are:

1. The robot can only move forwards (i.e. $v \geq 0$)

2. The robot rotates in perfect circular motion (i.e. no skidding). This is a valid assumption considering that the motion of the vehicles can be adequately described using a kinematic motion model.

3. The linear kinetic energy per unit mass e_l of the robot (Smith 2010) is given by

$$e_l = \frac{v^2}{2}. \tag{5.11}$$

   The rotational kinetic energy per unit mass e_r of the robot (Smith 2010) is given by

$$e_r = \frac{\pi \times w^2}{4}. \tag{5.12}$$

   Note that for simplification, Equation (5.12) assumes the moment of inertia of a disk rotating around its centre (i.e. $I = \frac{m \times R^2}{2}$, where m is the mass of the robot and R is its radius). This simplification can result in a slight mismatch between the desired angular velocity outputted by the controller and the maximum angular velocity that the robot can achieve. Nevertheless, it is expected that this effect will be relatively insignificant for most robotic applications. A more accurate model would use a more accurate description of the moment of inertia depending on the shape of the robot and its current turning radius.

4. The total kinetic energy per unit mass e_t of the robot is equal to the sum of its linear and rotational kinetic energies ($e_t = e_l + e_r$).

5. The maximum total kinetic energy per unit mass e_{max} is equal to the linear kinetic energy at maximum linear velocity v_{max}

$$e_{max} = \frac{(v_{max})^2}{2},$$

6. The maximum angular velocity w_{max} of the robot can be therefore found using

$$w_{max} = \sqrt{\left(\frac{4 \times e_{max}}{\pi}\right)}.$$

   Furthermore, if the angular velocity per timestep of the robot is too large, it can cause it to over-rotate around itself before it has the chance to re-examine its environment. This can significantly limit the control of the robot and its reaction to its surroundings. Therefore, if $w_{max} > \frac{\pi}{2 \times \Delta t}$, then w_{max} should be limited to $\frac{\pi}{2 \times \Delta t}$. In this way, the robot is limited to a maximum rotation of $\frac{\pi}{2}$ radians during a single timestep.

If it is detected that $e_t > e_{max}$, both the linear velocity v and the angular velocity w can be linearly decreased until $e_t = e_{max}$. In this way, if the PSO controller ends up requesting large linear and angular velocities that when combined exceed the maximum power output of the robot's actuators, it will be possible to linearly scale them down to match that correct power output.

5.2.1 Non-Omnidirectional PSO Velocity Update Equation

With the assumptions of the non-holonomic motion model in place, the PSO velocity update equation that will be used to control the velocity of the robots needs to be assembled. In Chapter 4, Adapted RPSO combined three tasks: 1) convergence to the personal best location of each robot, 2) convergence to the global best location and 3) obstacle avoidance. The algorithm consists of three different accelerating terms, one dedicated to each task. Therefore, to identify the number of accelerating terms that will be needed for the non-omnidirectional controller, the different tasks that the robot will need to carry out need to be identified. The different tasks that are taken into account by the equation are outlined below:

1. The angular velocity w is adjusted to rotate towards the personal best location $\tilde{\mathbf{y}}[k]$ (i.e. if $\tilde{\mathbf{y}}[k]$ is on the left of the robot, then w increases. Conversely, if $\tilde{\mathbf{y}}[k]$ is on the right of the robot, then w decreases).

2. Similarly, the angular velocity w is adjusted to face the global best location $\tilde{\mathbf{y}_\mathbf{g}}[k]$.

3. The angular velocity w decreases when there exists an obstacle (or another robot) on the left of the robot and the amount of decrease depends on the distance to the obstacle.

4. On the other hand, the angular velocity increases when there exists an obstacle

(or another robot) on the right of the robot. The amount of increase depends on the distance to the obstacle.

5. The linear velocity v of the robot is always increased. The amount of increase is adjusted to prevent collision with obstacles in front of the robot. As long as the linear velocity is limited properly, collision avoidance can be ensured no matter what the angular velocity is. If there are no obstacles, the linear velocity v is maximally increased.

From the outlined tasks, it can be seen that the PSO velocity update equation contains five accelerating terms. Four of them control the value of the angular velocity w and one controls the value of the linear velocity v. Also, notice that the last three tasks output a constant direction (i.e. the third task always decreases the angular velocity w, the fourth task always increases it and the fifth always increases the linear velocity v). The magnitude of the outputs of these tasks is adjusted (by the DVC strategy) but the direction (i.e. the output of the sgn functions in the accelerating terms) is always constant. It is only possible to use tasks in this way because of the use of a robot-centred reference frame, especially when it comes to the last task. If global reference frame Generalised Adapted PSO was used instead of Equation (5.4), it would have been impossible to command the robot to only move forwards, because the orientation of the robot is not taken into account by the velocity update equation.

Based on the previously described tasks, the PSO velocity update Equation (5.4) takes the following form

$$\tilde{\mathbf{u}}[k+1] = \boldsymbol{\omega} \circ \tilde{\mathbf{u}}[k] + \mathbf{c_1} \circ \mathbf{r_1} \circ \text{sgn}(\tilde{\mathbf{y}}[k]) + \mathbf{c_2} \circ \mathbf{r_2} \circ \text{sgn}(\tilde{\mathbf{y}_\mathbf{g}}[k]) - \mathbf{c_3} \circ \mathbf{r_3} + \mathbf{c_4} \circ \mathbf{r_4} + \mathbf{c_5} \circ \mathbf{r_5}, \quad (5.13)$$

where

$$\mathbf{c_1} = \begin{bmatrix} 0 \\ c_1 \end{bmatrix}, \quad \mathbf{c_2} = \begin{bmatrix} 0 \\ c_2 \end{bmatrix}, \quad \mathbf{c_3} = \begin{bmatrix} 0 \\ c_3 \end{bmatrix}, \quad \mathbf{c_4} = \begin{bmatrix} 0 \\ c_4 \end{bmatrix}, \quad \mathbf{c_5} = \begin{bmatrix} c_5 \\ 0 \end{bmatrix}.$$

The first four accelerating coefficient vectors (i.e. $\mathbf{c_1}$ to $\mathbf{c_4}$) only affect the angular velocity of the robot and therefore their first component is 0. Similarly, the last accelerating coefficient vector $\mathbf{c_5}$ only controls the linear velocity of the robot and therefore its second component is 0. The vector $\tilde{\mathbf{y}}$ is the position of the personal best location relative to the robot and $\tilde{\mathbf{y}_\mathbf{g}}$ is the position of the global best location relative to the robot. As described above, the last three accelerating terms do not contain a sgn function since their direction is always constant.

5.2.2 Dynamic Velocity Control Strategy for Non-Holonomic Robots

With the PSO velocity update equation assembled, it is now possible to define the DVC strategy that will be used. In a similar manner to Chapter 4, the DVC strategy presented here will aim to ensure that a single robot will be able, at any point, to avoid collision with its surroundings, while still being able to converge towards its personal best and global best locations. As long as this is true, the DVC strategy will be scalable to any number of robots.

As in Chapter 4, the proposed DVC strategy will only make use of maximum desired velocities. Therefore, instead of using a maximum desired acceleration A^+ or deceleration A^- to calculate $\hat{c}$, it will be calculated using

$$\hat{c} = \frac{\beta \times U}{\sqrt{d}}. \tag{5.14}$$

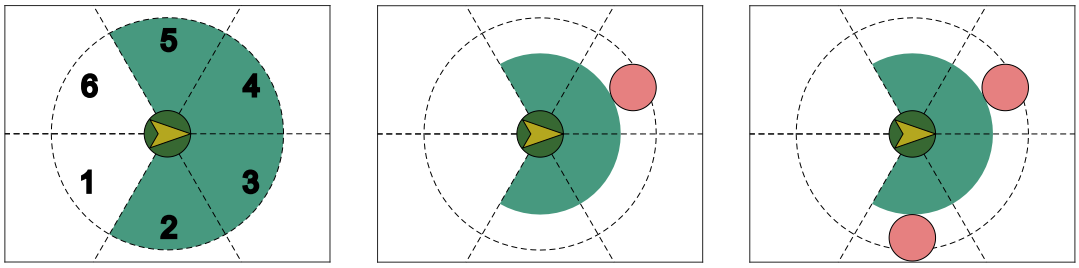
In Chapter 4, the maximum desired velocity was constructed using two velocity dimensions and therefore $d = 2$ was used. In this chapter, the linear velocity and angular velocity are calculated independently and therefore $d = 1$ will be used in each case.

Firstly, as in Chapter 4, the surroundings of the robot are separated into six sensing regions as shown in Figure 5-3a and only the closest body in each region is taken into account to approximate the operation of a lidar sensor. Regions 1, 2 and 3 are always on the right of the robot and regions 4, 5 and 6 are always on the left. Furthermore, $\mathbf{s}$ is the list of distances to the nearest obstacles, sorted in order from smallest to largest, such that s_1 is the distance to the closest obstacle. Also, let $\mathbf{sr}$ be the sorted list of the distances to obstacles on the right of the robot (regions 1-3), $\mathbf{sl}$ be the sorted list of the distances to obstacles on the left of the robot (regions 4-6) and $\mathbf{sf}$ be the sorted list of distances to the obstacles at the front of the robot (regions 2-5). These will be used to limit the parameters c_3, c_4 and c_5 respectively. Lastly, let

$$\boldsymbol{\nu} = \frac{\mathbf{sf}}{\Delta t}, \tag{5.15}$$

be a vector of speeds, such that $v = \nu_1$ is the minimum speed required for the robot to collide with the closest obstacle in front of it in the next timestep.

Angular Velocity: The next step is to calculate the amount of angular velocity that will be dedicated towards each term of the PSO velocity update equation. When no obstacles are close to the robot, the values of c_1 and c_2 should be maximised to allow the robot to turn towards the personal and global best locations. As an obstacle gets



(a) (b) (c)

Figure 5-3: Schematics that explain the DVC tuning strategy for the control of non-holonomic vehicles. The vehicle is described by the centred circle, where the arrow describes its orientation. The numbered areas represent the sensing regions of the vehicle and the circular dashed line represents its maximum sensing range. The red circles represent obstacles. In (a), the sensing regions are numbered. In (b), one obstacle is detected on the front left of the robot. This will cause a decrease in the linear velocity (represented by a decrease in the green region) and it will cause the robot to rotate rightwards. Furthermore, the linear velocity of the robot will be limited. In (c), There exist two obstacles, one on the left and one on the right, at an equal distance from the robot. This will limit the linear velocity of the robot but it will not cause any rotation.

closer to the robot, c_1 and c_2 should be decreased and c_3 and c_4 should be increased depending on the side where the obstacle is. This is achieved using

$$W_{1,2} = \frac{s_1}{R} \times w_{max} \qquad W_3 = (1 - \frac{sl_1}{R}) \times w_{max} \qquad W_4 = (1 - \frac{sr_1}{R}) \times w_{max} \tag{5.16}$$

where $W_{1,2}$ is the desired maximum angular velocity dedicated to the first two accelerating terms of Equation (5.13), W_3 is the desired maximum angular velocity dedicated to the third accelerating term, W_4 is the desired maximum angular velocity dedicated to the fourth accelerating term and R is the maximum sensing range of the robot. Note that if obstacles exist both on the left and right of the robot at equal distances from it, $W_3 = W_4$ and the third and fourth accelerating terms will tend to cancel each other out, likely causing no rotation due to obstacle avoidance. This can allow the robot to pass through small openings. Furthermore, no angular velocity is dedicated to the fifth accelerating term, since it is not meant to have any effect on the angular velocity of the robot. This procedure is described in Figures 5-3b and 5-3c.

From here, the parameters c_1 to c_4 can be calculated using Equation (5.14), where the corresponding desired maximum angular velocity W value will be used instead of U. The process of calculating c_1 and c_2 is as follows (where $\hat{c} = c_1 + c_2$):

- Using (5.14), calculate $\hat{c} = \beta \times W_{1,2}$, where $d = 1$. Based on the amount of

Algorithm 2: Swarm Control using Non-Omnidirectional Controller

foreach *robot* **do**
  robot.UpdatePersonalBestLocation(source_position);
  if *robot.f* $< f_g$ **then**
    $f_g \leftarrow$ robot.f; // Update global best location fitness
    $\mathbf{y_g} \leftarrow$ robot.$\mathbf{y}$; // Update global best location
  end
end
foreach *robot* **do**
  $s[] \leftarrow$ robot.GetDistanceToSurroundings(6); // 6 sensing regions
  $sr[] \leftarrow s[1:3]$; $sl[] \leftarrow s[4:6]$; $sf[] \leftarrow s[2:5]$;
  $W_{1,2} \leftarrow \min(s[])/R$; $W_3 \leftarrow 1 - \min(sl[])/R$; $W_4 \leftarrow 1 - \min(sr[])/R$;
  $\hat{c} \leftarrow \beta \times W_{1,2}$; // $\hat{c} = c_1 + c_2$
  $c_1 \leftarrow \hat{c}/2$; $c_2 \leftarrow \hat{c}/2$; // In this case $c_1/c_2 = 1$
  $c_3 \leftarrow \beta \times W_3$; // $\hat{c} = c_3$
  $c_4 \leftarrow \times W_4$; // $\hat{c} = c_4$
  // Update angular velocity
  robot.$w \leftarrow \omega *$ robot.$w + c_1 * rand() *$ sgn(robot.$\mathbf{y}_2$ − robot.$\mathbf{x}_2$) $+ c_2 * rand() *$ sgn(robot.$\mathbf{y_g}_2$ − robot.$\mathbf{x}_2$) $- c_3 * rand() + c_4 * rand()$;
  $\nu[] \leftarrow sf[].sort * \Delta t$; // Minimum velocity to collide with forward obstacles
  $U \leftarrow \alpha \times \nu[1]$;
  $c_5 \leftarrow \beta \times U$;
  robot.$v \leftarrow \omega *$ robot.$v + c_5 * rand()$; // Update linear velocity
end

Function *UpdatePersonalBestLocation(self, source_position)* **:**
  self.$\mathbf{x} \leftarrow$ self.GetCurrentPosition();
  $f =$ |self.$\mathbf{x}$−source_position|; // Assign distance-based cost to location
  if $d <$ *self.g* **then** // Update personal best location
    self.$f \leftarrow f$;
    self.$\mathbf{y} \leftarrow$ self.$\mathbf{x}$;
  end
end

angular velocity that has been dedicated to the first two terms (from (5.16)), the coefficients c_1 and c_2 can be calculated.

- Using $\hat{c} = c_1 + c_2$, calculate c_1 and c_2 based on the desired ratio $\frac{c_1}{c_2}$.

Similarly, c_3 is calculated as follows (note that in this case $\hat{c} = c_3$ and therefore the previous two steps are merged):

- Using (5.14), calculate $c_3 = \beta \times W_3$, where $d = 1$.

Repeat the last step using W_4 instead of W_3 to calculate c_4.

Linear Velocity: Finally, c_5 is adjusted based on the distance to the closest obstacle that is in front of the robot (i.e. using the list **sf**). The process of calculating c_5 is as follows (where $\hat{c} = c_5$):

- Set the desired maximum speed $U = \alpha \times \nu_1$.
- Using (5.14), calculate $c_5 = \beta \times U$, where $d = 1$.

The adjustment of c_5 is indicated in Figures 5-3b and 5-3c by the radius of the blue regions. The overall operation of the non-omnidirectional PSO controller with the DVC strategy described in this section is shown in Algorithm 2, where f refers to the fitness of the personal best location of a robot and f_g refers to the fitness of the global best location.

With the DVC strategy now defined, the ability of the PSO controller to control the motion of a swarm of differential drive robots will be tested. The following section will describe the MATLAB simulations performed, which are similar to the simulations presented in Chapter 4. Then the resulting behaviours of the two PSO controllers (omnidirectional PSO controller and non-omnidirectional PSO controller) will be compared.

5.3 Simulations and Results

To study the behaviour of a swarm when controlled by the non-omnidirectional PSO controller, 2D simulations in MATLAB were created. These simulations, aim to validate the use of the proposed non-omnidirectional PSO velocity update equation Equation (5.13) and the proposed DVC strategy. The simulation environment and setup was identical to the one used in Chapter 4, where the robots' velocities were directly controlled by the Adapted PSO controller. Since this simulation setup was already validated with Gazebo simulations in Chapter 4, this chapter will assume that the presented

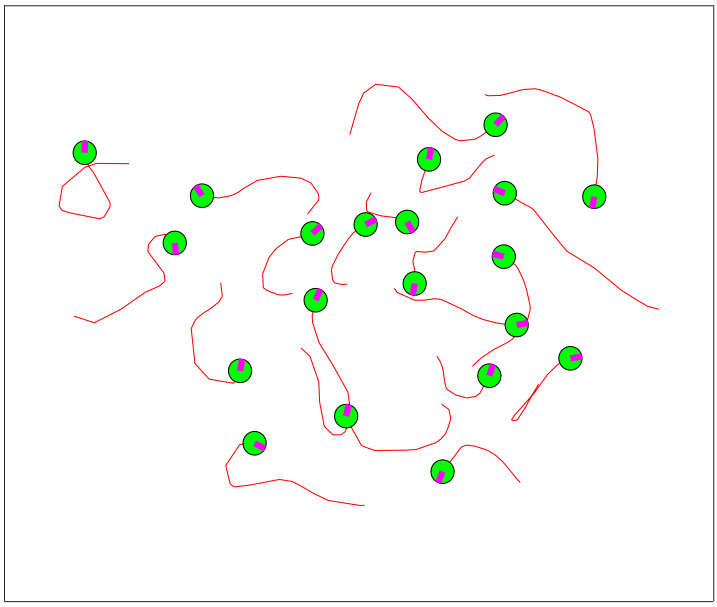


Figure 5-4: A simulated swarm of 20 robots (green circles) that are controlled by the non-omnidirectional PSO controller. The red lines indicate the past positions of each robot in the last 15 seconds. The robots never stop moving while no collisions occur between them.

MATLAB simulations do not need further validation with a realistic physics engine.

Initially, a swarm of 20 robots was run in an empty-space environment, to visually observe how the robots of a fairly large swarm interact with each other in the absence of obstacles. In the simulation, the robots appear to continuously move, while avoiding each other. The continuous forward thrust that is applied to them by the non-omnidirectional PSO controller, does not allow them to settle in one place and therefore they keep roaming, even after the swarm has reached the location of the source. Despite this seemingly chaotic behaviour, no collisions occur between robots. An example instance of the simulated swarm is shown in Figure 5-4.

5.3.1 Obstacle course simulations

To test the convergence rate and obstacle avoidance capabilities of the non-omnidirectional PSO controller, the following 2D simulations were performed, using an obstacle course. The simulation setup used, is identical to the one used in Chapter 4, to allow direct comparison between the non-omnidirectional PSO controller and the omnidirectional PSO controller (Adapted RPSO). As in Chapter 4, the obstacle course is described by Figure 4-1. The robots are assumed to be modified models of the Summit XL Steel platform (Robotnik Automation S.L.L. n.d.) of diameter 1 m and maximum

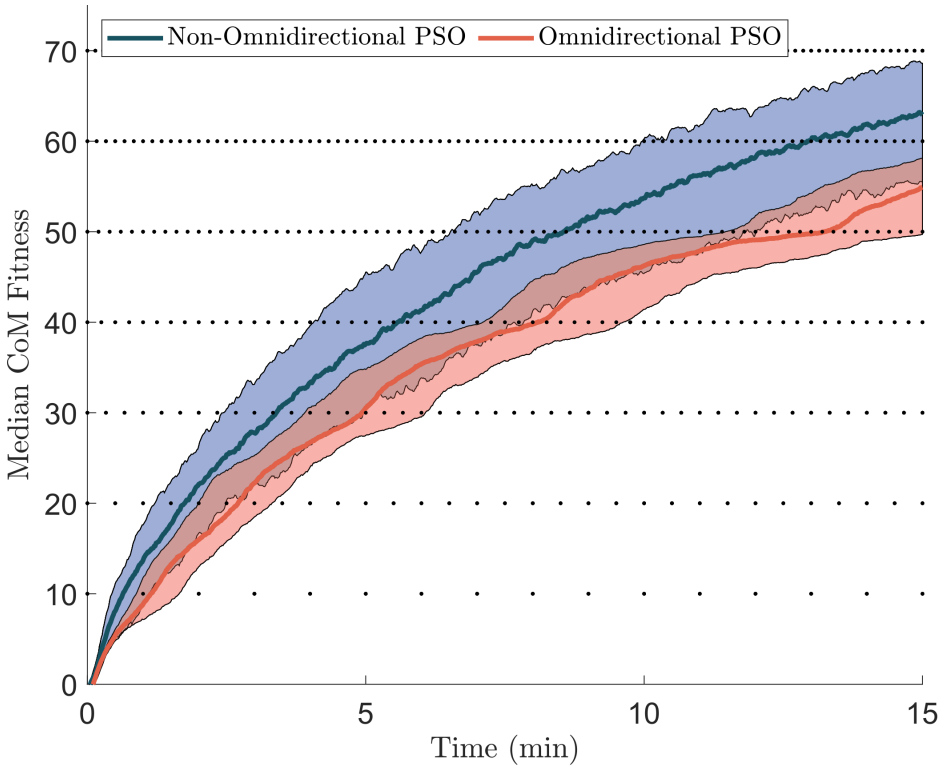


Figure 5-5: Median CoM fitness over time results for the non-omnidirectional and the omnidirectional PSO controllers. The transparent areas also show the 5^{th} and 95^{th} percentile CoM fitnesses of each controller. The dotted lines represent the obstacle layers of the obstacle course.

speed 3 m/s (the same model used in Chapter 4). Furthermore, as in Chapter 4, they are assumed to be equipped with LiDAR sensors and their obstacle detection range is limited to 3 m.

The parameter values used were also identical to allow direct comparison between the algorithms, as shown in Table 5.1. Note that the non-omnidirectional PSO controller does not have parameter γ assigned because it does not use the definition of virtual forces presented in Chapter 4 to achieve obstacle avoidance.

Table 5.1: Table of values used for different parameters of the two PSO controllers for the obstacle course simulations.

Controller	Case	α	β	γ	Δt
Omnidirectional PSO	DVC	0.9	0.9	1	1
Non-Omnidirectional PSO	DVC	0.9	0.9	-	1

Figure 5-5 shows the median results over 100 simulations for both the non-omnidirectional and the omnidirectional PSO controllers (the number of simulations and the number of robots in the swarm were the same as in Chapter 4 to allow direct comparison). The results of the omnidirectional PSO controller are identical to the results presented in Chapter 4 and they are included here for the purpose of comparison.

The results of Figure 5-5 show that the non-omnidirectional PSO controller allows the swarm to travel through the obstacle course faster than the omnidirectional PSO controller. This is understood to be a result of the continuous forward thrust that characterises the proposed non-omnidirectional PSO controller, which allows the robots to explore faster and progress further into the obstacle course on their own, as explained at the beginning of Section 5.2. From individual simulations, it was observed that when a robot passes through an obstacle layer first, it can continue exploring on its own, without having to wait for the rest of the swarm to push it through the next layer. Furthermore, in contrast to the omnidirectional PSO controller, the results of the non-omnidirectional PSO controller do not exhibit a clear trend of slowing down before passing through an obstacle layer. This does not mean that the robots do not slow down before passing through a layer (from individual simulations it was observed that they actually slow down significantly). Instead, due to the seemingly-chaotic movement of the robots, these trends are lost during the averaging process. Other signs of this seemingly-chaotic behaviour are the lines of the 5^{th} percentile, median and 95^{th} percentile lines, which are much noisier than the omnidirectional PSO controller, as well as the larger variance between these lines.

5.4 Non-omnidirectional PSO Controller with Minimum Turning Radius

There exist several different types of drives that can be characterised as non-holonomic. The most popular ones are the differential drive and traditional (or Ackermann) steering (Rubio et al. 2019, Bräunl 2008). The main way that traditional steering differs from differential drive, in terms of its resulting motion, is that it has a minimum turning radius. Therefore, control of traditional steering can be achieved using the non-omnidirectional PSO controller, as long as the turning radius that is requested by the PSO controller is not smaller than the minimum turning radius r_{min} of the vehicle. The turning radius r requested by the controller can be calculated using Equation (5.6). If the resulting r is smaller than r_{min}, then either v needs to be in-

creased or w needs to be decreased until $r = r_{min}$. That said, v is limited by the DVC strategy to avoid collisions and increasing it may counter this measure. Therefore, the angular velocity w needs to be decreased, until $r = r_{min}$ is achieved.

At first sight, this appears to be a straightforward modification, but it introduces significant complexity into the PSO controller. The ability of a differential drive robot (i.e. no minimum turning radius) to rotate in place, allows it to avoid obstacles, even when they are right in front of it, without having to move backwards. In the case of traditional steering though, this is not possible. This problem can be avoided, by allowing the robot enough space to rotate, by setting up a minimum distance that the robot can have from an obstacle. For example if the minimum turning radius of a robot is $r_{min} = 0.5\,\text{m}$, the robot can be forced to maintain a minimum distance $s_{min} > 1\,\text{m}$ from obstacles (i.e. robot-obstacle separation), allowing it enough room to fully rotate and move away. That said, it is not clear at this point, what the actual mathematical relationship between s_{min} and r_{min} needs to be to ensure no collisions.

The introduction of the minimum turning radius r_{min} can allow the non-omnidirectional controller to approximate the motion of a large number of vehicles, like car-type land vehicles and marine vehicles like ASVs. Additionally, Equation (5.4) can be adapted for use in 3D environments through the introduction of an additional angular velocity term (representing the rate of change of the pitch of the robot). This can allow the controller to be used in underwater vehicles like AUVs and forward flight aircraft. Nonetheless, further study needs to be performed to identify the relationship between s_{min} and r_{min}. For the rest of this thesis, unless it is otherwise specified, the non-omnidirectional controller with no minimum turning radius, as described throughout this chapter, will be used for the control of simulated ASVs, since it most closely approximates their motion.

5.5 Conclusions

This section has introduced a new approach to the implementation of PSO for robotic applications. The use of a robot-centred reference frame for the updating of the velocity vector overcomes some significant problems of previous versions of PSO. Furthermore, it allows the implementation of non-omnidirectional PSO controllers, other than the traditional omnidirectional controller that PSO is typically associated with. A non-omnidirectional PSO controller and DVC strategy are also introduced. The non-omnidirectional controller is compared to the omnidirectional controller for the task of navigating through an obstacle course and exhibits better performance but also

more seemingly chaotic motion. The non-omnidirectional controller is an important extension of the Generalised Adapted PSO algorithm because it allows the control of a larger number of robotic platforms, while considering the movement limitations of each robot (beyond maximum velocity and acceleration). A further adaptation to a non-omnidirectional PSO controller with minimum turning radius is also discussed.

References

Batlle, J., Nicosevici, T., Garcia, R. & Carreras, M. (2003), 'ROV-Aided Dam Inspection: Practical Results', *IFAC Proceedings Volumes* **36**(21), 271–274.
URL: *https://www.sciencedirect.com/science/article/pii/S1474667017378199*

Bräunl, T. (2008), *Embedded Robotics: Mobile Robot Design and Applications with Embedded Systems*.

Choi, J., Lee, Y., Kim, T., Jung, J. & Choi, H.-T. (2017), Development of a ROV for visual inspection of harbor structures, *in* '2017 IEEE Underwater Technology (UT)', pp. 1–4.

Figueiredo, A. B., Ferreira, B. M. & Matos, A. C. (2014), Tracking of an underwater visual target with an autonomous surface vehicle, *in* '2014 Oceans - St. John's', pp. 1–5.

Janson, S. & Middendorf, M. (2007), On Trajectories of Particles in PSO, *in* '2007 IEEE Swarm Intelligence Symposium', pp. 150–155.

Ludvigsen, M., Johnsen, G., Lagstad, P., Sørensen, A. & Odegard, O. (2013), Scientific Operations Combining ROV and AUV in the Trondheim Fjord, *in* 'Marine Technology Society Journal', Vol. 48, pp. 1–7.

Macreadie, P. I., McLean, D. L., Thomson, P. G., Partridge, J. C., Jones, D. O. B., Gates, A. R., Benfield, M. C., Collin, S. P., Booth, D. J., Smith, L. L., Techera, E., Skropeta, D., Horton, T., Pattiaratchi, C., Bond, T. & Fowler, A. M. (2018), 'Eyes in the sea: Unlocking the mysteries of the ocean using industrial, remotely operated vehicles (ROVs).', *The Science of the total environment* **634**, 1077–1091.

Nad, D., Miskovic, N. & Mandic, F. (2015), 'Navigation, guidance and control of an overactuated marine surface vehicle', *Annual Reviews in Control* **40**, 172–181.
URL: *https://www.sciencedirect.com/science/article/pii/S1367578815000474*

Njaka, T., Brizzolara, S. & Ben-Tzvi, P. (2020), 'Design and Experimental Validation of a Novel High-Speed Omnidirectional Underwater Propulsion Mechanism', *IEEE/ASME Transactions on Mechatronics* p. 1.

Robbins, I. C., Kirkpatrick, G. J., Blackwell, S. M., Hillier, J., Knight, C. A. & Moline, M. A. (2006), 'Improved monitoring of HABs using autonomous underwater vehicles (AUV)', *Harmful Algae* **5**(6), 749–761.
URL: *https://www.sciencedirect.com/science/article/pii/S1568988306000370*

Robotnik Automation S.L.L. (n.d.), 'SUMMIT-XL STEEL MOBILE ROBOT'.
URL: *https://robotnik.eu/products/mobile-robots/summit-xl-steel-en/*

Rubio, F., Valero, F. & Llopis-Albert, C. (2019), 'A review of mobile robots: Concepts, methods, theoretical framework, and applications', *International Journal of Advanced Robotic Systems* **16**(2), 1729881419839596.
URL: *https://doi.org/10.1177/1729881419839596*

Smith, J. O. (2010), *Physical Audio Signal Processing*, W3K Publishing.

Spears, W., Green, D. & Spears, D. (2010), 'Biases in Particle Swarm Optimization.', *IJSIR* **1**, 34–57.

Chapter 6

Marine Acoustics and Acoustic Signal Processing

The previous chapters have focused on the adaptation of the PSO algorithm to offer precise motion control for robotic swarms and allow its merging with other swarm robotic techniques. That said, the main purpose of the algorithm in this thesis, which is the localisation of marine acoustic sources, has not been studied yet. The next chapters will focus on this task, but first it is important to present the individual characteristics that define underwater acoustic signals and differentiate them from other signals found in nature. This chapter will present a review of the literature, discuss a number of propagation models for underwater acoustic signals and present the most significant acoustic sources that can be found in marine environments. Afterwards, the basic principles of acoustic signal processing techniques will be introduced and it will be described how they can be linked to swarm robotics.

6.1 Underwater Sound Propagation

Sound propagation in the sea has been studied extensively. Field measurements relative to a large variety of parameters have been performed in different oceanic areas, enabling the implementation of accurate modelling software (Richardson et al. 2013). However, these types of models are usually application specific and are not easily generalised. Instead, there exist several simpler general models that can be used to estimate the transmission loss of a propagating underwater acoustic signal for different cases. These models will be presented in this section.

The simplest model that can be used to approximate the transmission loss of an underwater acoustic signal at a distance R, is the *spherical spreading loss* model (Rossing & Fletcher 2004)

$$L = L_s - 20\log_{10}(R) - 60 \tag{6.1}$$

where: L is the received sound pressure level (SPL) at range R (km) from the source, in dB re 1µPa,
L_s is the source level in dB re 1µPa·m and
-60 is a conversion factor related to the change in range units from L_s at 1 m to L at R km from the source.

The spherical spreading model can be used to approximate the transmission loss of a signal as it propagates in deep waters (i.e. when R is smaller than the ocean depth), as shown in Figure 6-1a. For shallow waters (i.e. when R is larger than the ocean depth), as shown in Figure 6-1b, this model is adjusted to use cylindrical propagation, resulting in the *cylindrical spreading loss* model (Rossing & Fletcher 2004)

$$L = L_s - 10\log_{10}(R) - 60 \tag{6.2}$$

Apart from the geometric spreading loss, an additional absorption loss term can also be included (Rossing & Fletcher 2004, Triwahyanti et al. 2018), resulting in

$$L = L_s - 20\log_{10}R - \alpha R - 60 \tag{6.3}$$

and

$$L = L_s - 10\log_{10}R - \alpha R - 60, \tag{6.4}$$

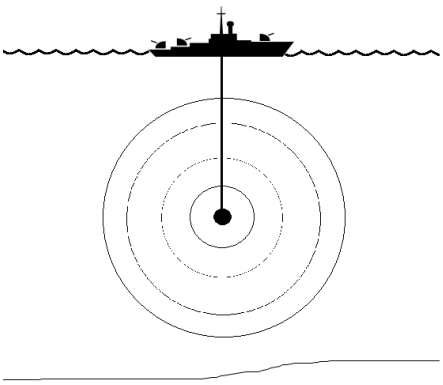
where: α is the sound absorption coefficient in dB/km.

The value of α is given by

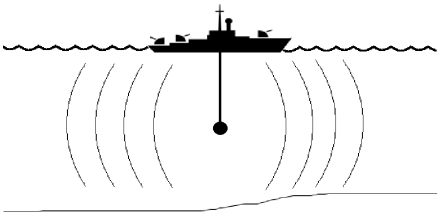
$$\alpha = \alpha_0\ \omega^{\eta}, \tag{6.5}$$

where ω is the angular frequency of the signal and α_0 and η are parameters that depend on the propagation medium (e.g. oceanic water). For underwater acoustic signals, the sound absorption coefficient α is usually about 0.001 dB/km at 100 Hz signal frequency and 1 dB/km at 10 kHz (Rossing & Fletcher 2004). This is much smaller than the corresponding sound absorption coefficient values for air, which is the reason that sound propagates so efficiently in water.

Finally, Equations (6.3) and (6.4) can be merged together into a single equation which employs spherical spreading up to a range R_1 and cylindrical spreading beyond that



(a) Spherical Spreading



(b) Cylindrical Spreading

Figure 6-1: Schematics that describe the two main types of signal propagation, using a ship-mounted SONAR as an acoustic source example.

range (Richardson et al. 2013)

$$L = L_s - 10\log R_1 - 10\log R - \alpha R - AR - 60 \tag{6.6}$$

where: R_1 is the range in km where the transition from spherical to cylindrical spreading occurs and
A is the transmission loss anomaly in dB/km.

The transmission loss anomaly A is used to represent additional energy losses due to reflection from the surface and the bottom of the ocean, as well as changes in the propagation due to variations in the temperature, depth and salinity of the oceanic water. This is an empirical term that is usually tuned based on collected data for each scenario (Federation of American Scientists 1998).

The work described in this thesis is largely theoretical and simulation-based and the

presented concepts aim to be as generalised as possible to allow easy adaptation to different scenarios. Despite the increasing accuracy of the models described by Equations (6.3), (6.4) and (6.6), they also become application specific. Therefore, the rest of this thesis will make use of the more general spherical spreading loss model of Equation (6.1), while more accurate models can be used in future studies as the introduced concepts are applied to specific scenarios, where the values of R, α, A etc. are defined accordingly.

6.2 Marine Acoustic Sources

Section 6.1 described the physical characteristics of acoustic signal propagation in oceanic environments but did not present the different types of acoustic signals that can be encountered. This section will present the main types of sounds produced by various marine acoustic sources (both biological and anthropogenic), outlining specific important characteristics (e.g. intensity, frequency, etc.). There are three main categories of marine sounds (1) marine mammal sounds, (2) anthropogenic sounds and (3) ambient noise. Each will be described in the following sub-sections.

6.2.1 Marine Mammal Sounds

Marine mammals use underwater acoustics for the communication of information such as the presence of food, danger or another individual of the species. Furthermore, some species are capable of echolocation; that is the ability to acoustically locate an object and identify characteristics such as its size, shape and movement. There has been significant research on the sounds made by several species but there still exist many gaps in the literature, especially due to the wide range of sounds that each species produces and the large number of specific functions that each sound can correspond to. Furthermore, up until now it has been difficult to study some of these animals, primarily due to their large size and their inability to survive in captivity.

Sounds produced by marine mammals like tonal moans, pulses, songs, thumps in pairs, up-down frequency sweeps etc. are usually in the frequency range of approximately 20 Hz to 4 kHz. Clicks and whistles on the other hand are emitted at higher frequencies, typically between 2 and 20 kHz (Würsig et al. 1993, Cummings & Thompson 1994, Watkins 1980, Richardson et al. 2013). The estimated source levels of all sounds are ranging from 109 to 192 dB re 1µPa·m (Thompson et al. 1986, Cummings & Holliday 1987, Richardson et al. 2013) depending on the species.

Sounds used in echolocation are of high intensity and frequency and of short duration,

typically between 50-200 µs. The pulses are emitted forwards in highly directional beams and there is usually a break between pulses to make sure that the echo from a pulse is received before the next one is produced. The extreme directionality of the signals allows the intensity of the signal to be higher than the signal of an omnidirectional transmitter of the same power. For this reason, among all recorded marine mammal sounds, echolocation signals have the highest source levels, ranging from approximately 120-230 dB re 1µPa·m. For some species the frequencies range between 20-60 kHz and 100-130 kHz in low and high ambient noise levels respectively, but they can go as low as 25 kHz and as high as 150 kHz (Au 1993).

6.2.2 Anthropogenic Sounds

Apart from biological marine acoustic sources, in the recent era, large quantities of anthropogenic noise have also been introduced into oceanic environments significantly altering the ocean's acoustic environment. Such sounds are created both purposefully and unintentionally and are typically found along well travelled paths in the sea, particularly in coastal and continental shelf waters (Hildebrand 2005).

The most common type of sources for anthropogenic sounds is means of transportation consisting of boats, small ships, commercial vessels and supertankers and other vehicles and their sound characteristics typically vary with the size of the vehicle. Although small boats are very common in coastal water, there is not much data about their acoustical characteristics. For this type of vessels, where the engines are usually outboard, noise level spectra contain strong tones at frequencies up to several hundred Hz (Richardson et al. 2013).

In contrast, small ships (characterised by lengths in the range 55 m to 85 m), typically emit sounds of fundamental frequencies around 10 Hz to 11 Hz (Greene Jr 1987). Broadband source levels for vessels of this category are ranging at about 170 dB re 1µPa·m to 180 dB re 1µPa·m (Hall et al. 1994). For larger ships (commercial vessels and supertankers), noise levels are highest at very low frequencies, typically around 180 dB re 1µPa·m and for some, they exceed 205 dB re 1µPa·m for frequencies as low as 2 Hz (Richardson et al. 2013).

6.2.3 Ambient Noise

Below the sea surface, noise tends to be of low intensity, compared to noise that can exist in some areas above the surface (e.g. cities). Nevertheless, the ambient noise that can be detected is still significant enough to require consideration during the design of

an underwater acoustic system. This sub-section will present the types and sources of underwater ambient noise.

Deep Water: Deep water regions represent around 90% of the area covered by oceans. In these areas, noise with frequency <20 Hz arises from seismic sources, turbulence, tides and waves and sounds of ship propeller blades. This low levels of frequencies are generally not affected by the wind and are the most difficult to measure due to the effect that turbulence and water currents can have on the equipment used. From 20 to 300 Hz, the dominant source of noise is distant shipping, with some minor additional effect from the wind and from 300 to 500 Hz, strong winds can produce noise that overshadows that of distant shipping. From 500 Hz to 50 kHz, ambient noise consists mainly of wind, waves and rain sounds. Higher frequencies are also greatly affected by thermal agitation (Urick 1983). Probably the most famous study for underwater ambient noise was performed by Wenz (1962) and its results are described by Figure 6-2. Despite being carried out in 1962, the study is still considered the most descriptive model of marine ambient noise (Rossing & Fletcher 2004).

Shallow Water: Shallow water regions are usually defined as regions that are <200 m deep. Sources of noise in shallow waters can be separated into three categories, (1) wind and waves (2) biological noise (3) distant shipping, industrial or seismic survey noise In these depths, ambient noise experiences a wider range of levels. Above 500 Hz, noise levels can be higher than deep water regions by about 5-10 dB. At he same time, below 300 Hz and especially with the lack of distant shipping and biological noise, the expected levels are lower than in deep waters. Wind also has a big effect on noise in these regions. At wind speeds of about 9 km/h, wind speed can predict ambient noise levels better than most of the other noise sources (Urick 1983).

6.2.4 Signal-to-Noise Ratio Calculation

Based on the information provided in Sections 6.2.1 to 6.2.3 and using the propagation models of Section 6.1, it is now possible to calculate the Signal-to-Noise Ratio (SNR) of signals emitted by different sources at a given distance from the source. This can allow the calculation of the maximum range of different source localisation systems (including swarm robotic systems), based on the type of source that they aim to localise.

To calculate the SNR of an observed signal, it is first needed to identify the average intensity of the noise. Figure 6-2 provides the power spectrum levels of different types of noise that can be encountered in the deep sea. Given a range of frequencies, the average

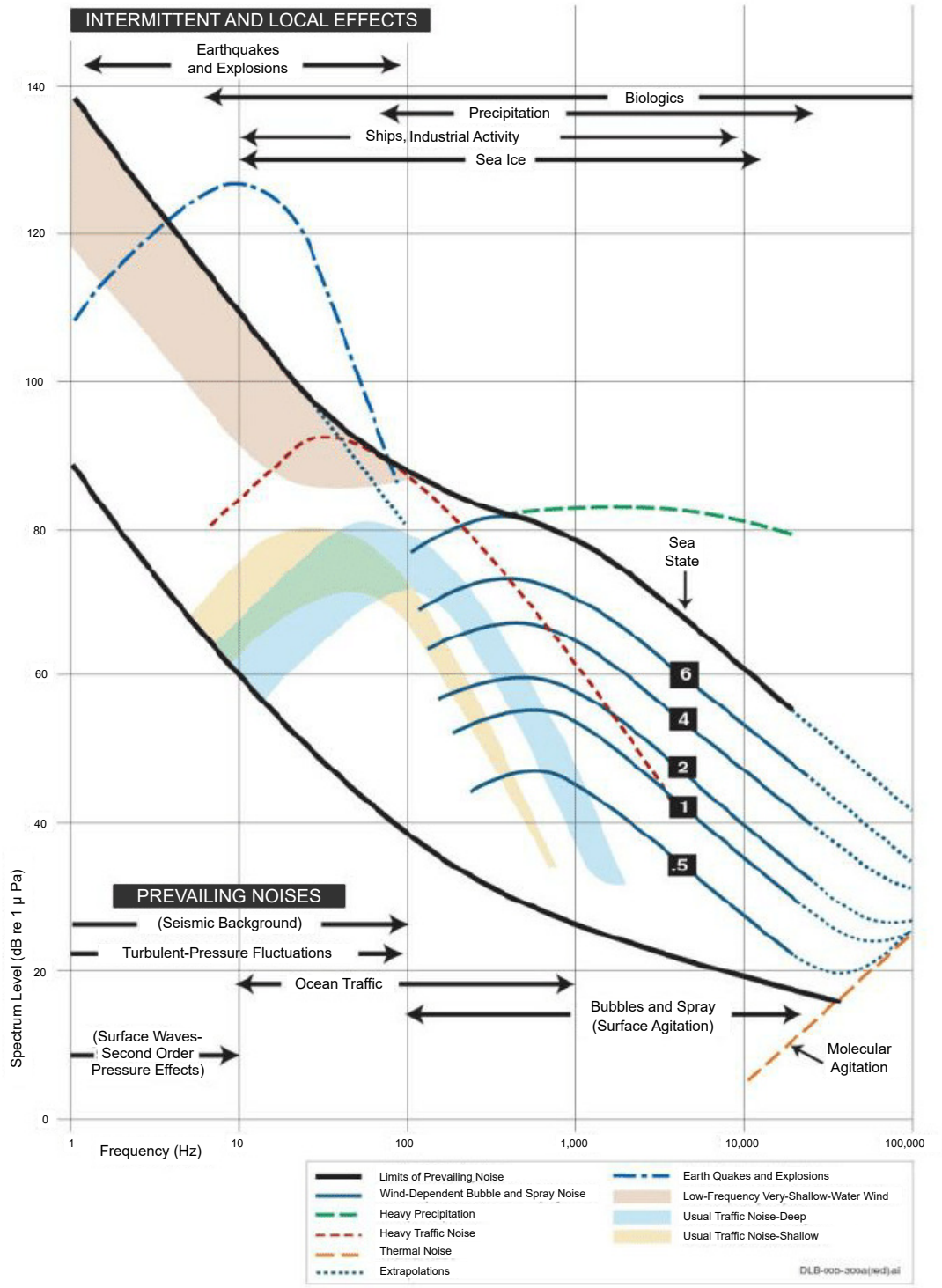


Figure 6-2: Composite of ambient-noise spectra, summarizing results and conclusions about spectrum shape, level, and probable sources of ambient noise between 1 Hz and 100kHz (Wenz 1962).

intensity level N of a selected type of noise is given by the area under the corresponding noise line in Figure 6-2. Assuming the propagation model of Equation (6.1), the SNR of a signal of source level L_s at distance R from the source is given by

$$\text{SNR} = L_s - 10\log_{10}\left(\frac{N}{10^{-6}}\right) - 20\log_{10}(R) - 60, \tag{6.7}$$

where the value 10^{-6} is the reference acoustic pressure of 1 µPa, used by convention in underwater acoustics.

Marine acoustic source localisation systems are highly dependent on the SNR of the received signals. When the SNR is very low, localisation may be impossible, effectively limiting the maximum localisation range of such systems. Therefore it is desirable to employ techniques that can improve the SNR of received signals, thereby improving the range and performance of such systems. In acoustic signals, such techniques are possible through the use of wavefield correlation (i.e. employing signal characteristics like frequency, bandwidth and propagation speed). The following section will describe the basic principles of such techniques and it will be explained how swarm robotic systems can be employed to extend their capabilities.

6.3 Wavefield Correlation

Sources in nature can be separated into two types: 1) wavefield sources (e.g. acoustic (Kundu 2014), electromagnetic (Hao et al. 2015) etc.) and 2) non-wavefield sources (e.g. chemical (Pomareda et al. 2017), radioactive (Gao et al. 2018) etc.). The main difference between the two types is that wavefield sources emit periodic signals. Therefore, while signals from non-wavefield sources are characterised mainly by their intensity, signals from wavefield sources have additional characteristics such as frequency and propagation speed. Due to these characteristics, wavefield signals contain additional information about the location of the source that can be extracted by cross-correlating simultaneous signal readings from multiple sensors. Additionally, techniques that rely solely on the intensity of the signal are generally more sensitive to low Signal-to-Noise-Ratio (SNR) than techniques that employ cross-correlation. This concept will be discussed in more detail in this section.

Due to the periodic nature of underwater acoustic signals, techniques that employ cross-correlation are applicable to a large number of marine applications such as seismography (Rost & Thomas 2002), seafloor mapping (Makowski & Finkl 2016) and imaging of the oceanic environment. Active SONAR systems emit acoustic signals that are reflected

from the surfaces of nearby objects (Stewart & Westerfield 1959, Fortmann et al. 1983) and cross-correlation techniques are used to measure the time-of-flight of the signal, thereby allowing imaging reconstruction of the oceanic environment. On the other hand, passive SONAR uses only the acoustic signals emitted by sources and cross-correlation is employed to identify their location.

A number of different techniques exist that employ cross-correlation - e.g. angle-of-arrival (AOA), time-of-arrival (TOA), time-difference-of-arrival (TDOA) etc., and all of them operate based on the same basic principles of cross-correlation (Munoz et al. 2009). This chapter will describe these principles, using a marine acoustic scenario and will outline their limitations.

6.3.1 The Two-Hydrophone Receiver Model

To introduce the basic principles of cross-correlation of acoustic signals, consider the simplest multi-hydrophone system configuration: the two-hydrophone receiver model (Li et al. 2019), located at a far field point relative to an acoustic source, as shown in Figure 6-3.

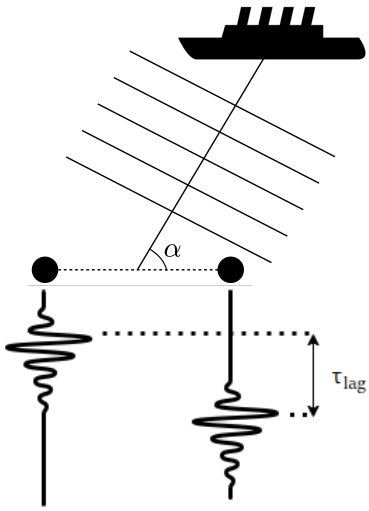


Figure 6-3: Schematic that describes how an incoming signal emitted by an external wavefield source (ship) is received by a pair of hydrophones (black circles). In this case, the signal is first received by the right-most hydrophone and after time τ_{lag}, it is received by the left-most one. The angle α is the AOA of the signal (i.e. direction towards the source).

Let $s_1(t)$ and $s_2(t)$ be the signal readings of the two hydrophones, which are assumed to be separated by a distance D. If the signal arrives at the hydrophones at an angle of

arrival (AOA) α, by the time that it reaches the first hydrophone, it needs to propagate an additional distance $D\cos(\alpha)$ before reaching the second one. Therefore, if the signal is assumed to propagate at a constant speed c, the time delay τ_{lag} between $s_1(t)$ and $s_2(t)$ is given by

$$\tau_{lag} = \frac{D\cos(\alpha)}{c}. \tag{6.8}$$

Alternatively, if the time delay τ_{lag} is known, the whole process can be reversed, to calculated the AOA of the signal (i.e. the direction towards the source), which can then be used to localise the source.

The time delay τ_{lag}, can be obtained by cross-correlating the two signals $s_1(t)$ and $s_2(t)$. Cross-correlation is the mathematical process used to describe the similarity between two signals and it is described by

$$R(\tau) = \int_{-\infty}^{\infty} s_1(t)s_2(t+\tau)dt \tag{6.9}$$

where τ is an arbitrary time delay added to $s_2(t)$. $R(\tau)$ is maximised at $\tau = \tau_{lag}$ such that

$$\tau_{lag} = \text{argmax}_{\tau}(R(\tau)). \tag{6.10}$$

From here, τ_{lag} can be used in Equation (6.8) to calculate the AOA of the received signal and thereby the direction towards the source.

Apart from measuring the AOA of a signal, cross-correlation can also be used to enhance the SNR of intensity measurements. The non-normalised correlation coefficient ρ is the value of $R(\tau)$ at $\tau = \tau_{lag}$

$$\rho = R(\tau_{lag}). \tag{6.11}$$

As the average intensity of $s_1(t)$ and $s_2(t)$ increases, ρ increases too. That said, ρ is less sensitive to noise and therefore, it can be used in place of intensity measurements to reduce the negative effects of noise in techniques that employ intensity measurements.

6.3.2 Ambiguities

Section 6.3.1 described how a two-hydrophone receiver can be implemented for the calculation of the AOA of a signal, through the use of cross-correlation. Despite its advantages, this model comes with several limitations that need to be taken into account to avoid the introduction of ambiguities in the calculation of the angle α. These ambiguities will be presented in this sub-section.

The first type of ambiguity occurs due to the use of only two hydrophones. The AOA

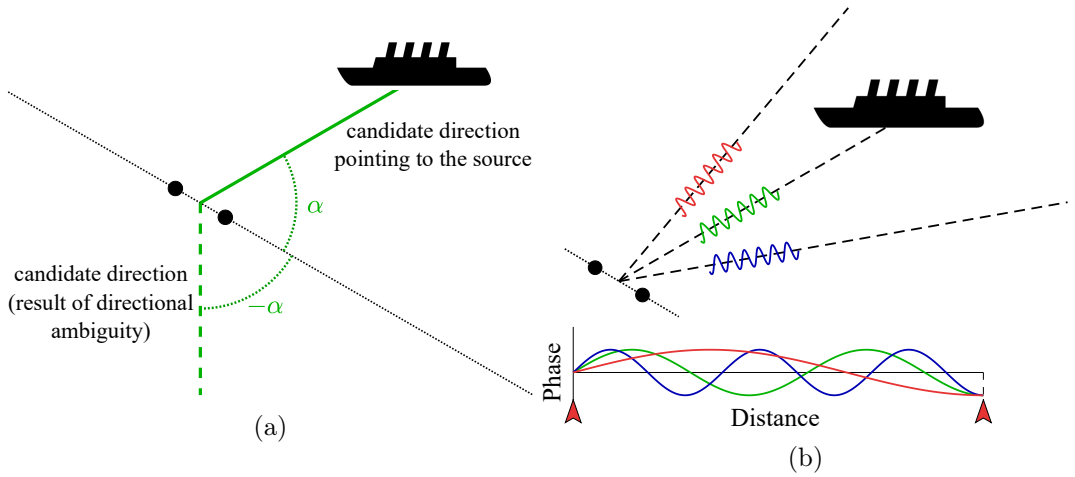


Figure 6-4: Schematics that describe the two problems of directional ambiguity, where the black circles represent the pair of hydrophones. In (a), the angle α is the calculated angle of arrival of the signal. The solid and dashed green lines are the leftward and rightward candidate directions towards the source respectively. In (b), the green (middle) signal represents the signal emitted by the source, while the red and blue (side) signals represent ambiguities caused by the hydrophone separation. The corresponding red, green and blue phase profiles show how all three signals can result in the same phase difference for the two hydrophones (arrows on the phase-distance graph).

of the signal is calculated with respect to the line formed by the two hydrophones. However, it is impossible to know if this angle is negative or positive, resulting in a directional ambiguity as shown in Figure 6-4a.

Additional ambiguities can be introduced due to the hydrophone separation D. The time delay τ_{lag} is calculated based on the phase difference $\Delta\phi$ of the signal readings of the two hydrophones, which is given by

$$\Delta\phi = \frac{2\pi f_c D\ \cos(\alpha)}{c} = \frac{2\pi D\ \cos(\alpha)}{\lambda_c}, \tag{6.12}$$

where f_c is the central frequency of the signal and λ_c is the corresponding wavelength of the signal.

When the received signal is narrowband, using Equation (6.9) it can be seen that the correlation between $s_1(t)$ and $s_2(t)$ drops as $|\Delta\phi|$ approaches π (i.e. the signal readings go out-of-phase). That said, when $|\Delta\phi| > \pi$, the two signal readings start going in-phase again and their correlation increases. This can result in the same phase difference when the signal arrives at the hydrophones from different angles, as shown in Figure 6-4b. Therefore, to avoid this type of ambiguity, the phase difference must be limited to

$|\Delta\phi| \leq \pi$, which can be achieved using $D \leq \lambda_c/2$ in Equation (6.12). Alternatively, as the bandwidth of the signal increases, these types of ambiguities are generally reduced, since components of larger wavelengths are introduced (Munoz et al. 2009).

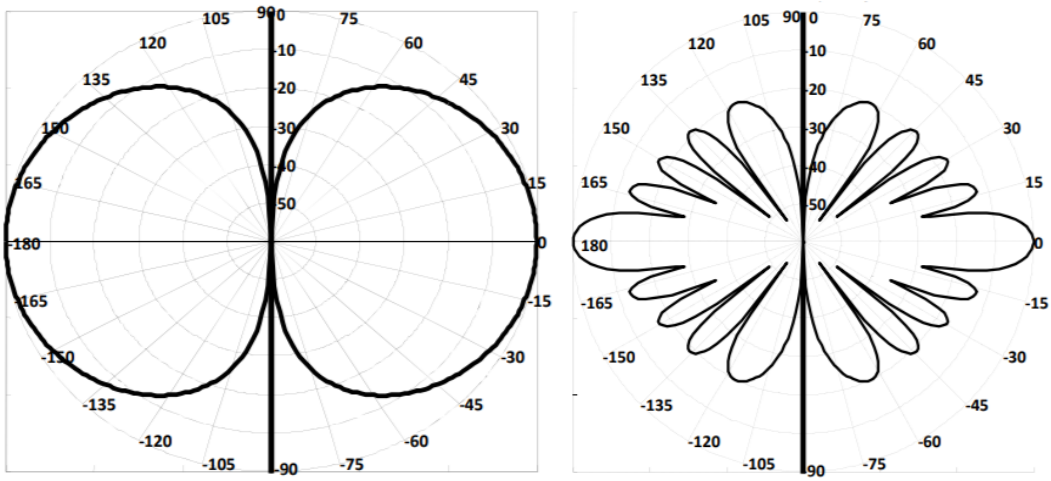
6.3.3 Extended Multi-Hydrophone Configurations

The two-hydrophone receiver model presented in this section can be extended to models that employ three or more hydrophones in different configurations, aiming to further increase the SNR of the signal readings and address the ambiguity problems of the two-hydrophone configuration. Several different types of configurations exist, each with its own advantages, which will be summarised in this sub-section.

Uniform linear n-hydrophone arrays: The two-hydrophone configuration can be extended to any number n of colinear hydrophones. This type of configuration is well-studied in underwater acoustics due to its relative operational simplicity (e.g. hydrophone arrays can be towed by single vessels (Miller & Tyack 1998, Yack et al. 2013)). Due to the colinear placement, the second type of ambiguity (i.e. described in Figure 6-4b) are reduced. This is shown in Figure 6-5, where the regions outside the lobes represent the regions where ambiguities occur. For example, for the two-hydrophone array of Figure 6-5a, ambiguities occur at -90^o and 90^o (i.e. when the direction of propagation of the received signal is parallel to the hydrophone array).

The main advantage of this configuration is that it can greatly boost the SNR of the signal readings, which in turn increases their maximum range of detection. Therefore, even though the ten-hydrophone array of Figure 6-5b may appear to introduce more ambiguity regions, in practice, its main lobes (i.e. 0^o and 180^o) correspond to greater sensitivity, compared to the two-hydrophone array of Figure 6-5a. Finally, it should be noted that, since the hydrophones are colinear, this configuration does not address the first type of ambiguities (i.e. described in Figure 6-4a).

Non-linear hydrophone distributions: This type of configuration can be achieved by adding one or more non-colinear hydrophones to the two-hydrophone receiver (Wang et al. 2019). The direct advantage of this configuration is that the extra hydrophones can be used to readily resolve ambiguities of the first type (Figure 6-4a). The second type of ambiguity(Figure 6-4b) is also addressed using this configuration, through the addition and proper placement of extra non-colinear hydrophones (Malgoezar et al. 2016) but the SNR boost may be smaller compared to the linear array configuration. Additionally, by treating individual pairs of hydrophones as individual two-hydrophone



(a) Hydrophone array with 2 elements (b) Hydrophone array with 10 elements

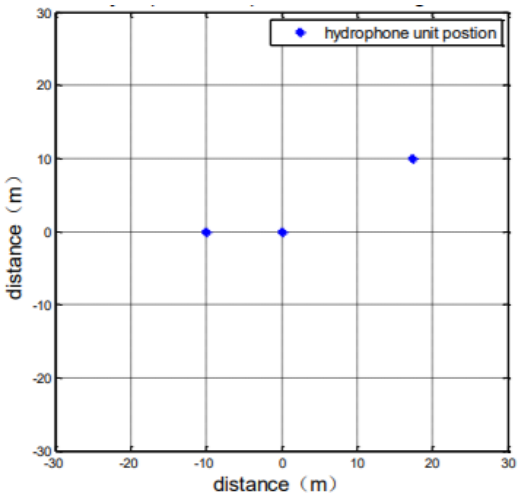
Figure 6-5: Normalised polar sensitivity pattern for hydrophone arrays with different number of elements. The hydrophone arrays are placed vertically (i.e. the -90^o to 90^o line) and consecutive hydrophones are separated by a distance of $\lambda/2$, where λ is the wavelength of the received signal. The figures were obtained using the MATLAB Phased Array System Toolbox.

receivers, localisation of the source can be achieved using triangulation. The effect of proper positioning for this task is shown in Figure 6-6, where the localisation accuracy of two different non-linear hydrophone configurations are compared (Wang et al. 2019).

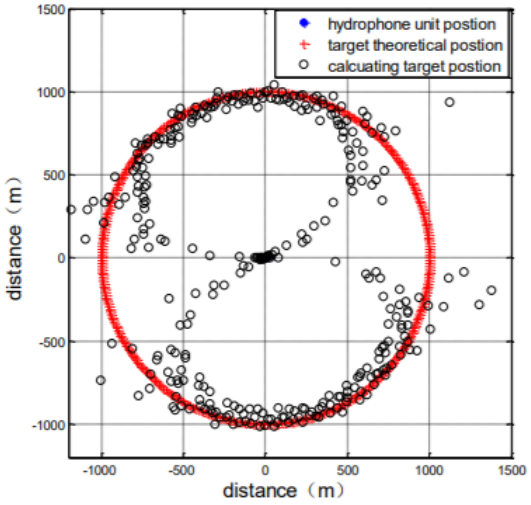
The two types of hydrophone configurations can be combined together, resulting in a variety of more complex configurations. Furthermore, recent studies have proposed the use of autonomous distributed hydrophone nodes capable of dynamically altering their positions in order to adaptively resolve ambiguities, based on the current needs of the system (Jiang et al. 2019). This has led researchers to consider the use of swarm robotic techniques for the control of such hydrophone nodes. The next chapter will describe how the wavefield correlation techniques presented in this section can be fused with the swarm control algorithms described in previous chapters to overcome some of their problems while also increasing their range and source localisation speed.

6.4 Conclusion

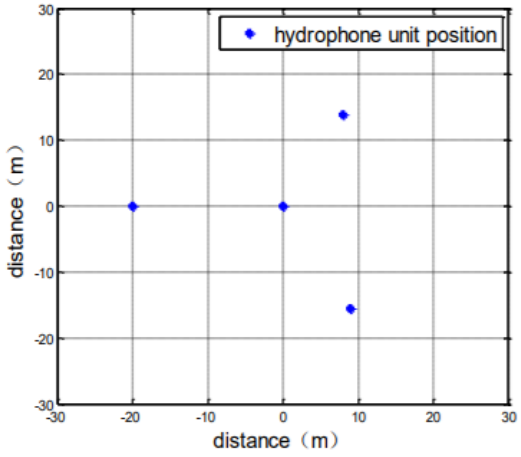
This chapter has presented the background literature, relevant to marine acoustics and wavefield correlation. The presented information and techniques will be used in the next chapters for the adaptation of PSO for source localisation applications. Sections 6.1 and 6.2 introduced a number of propagation models that can be used to estimate how



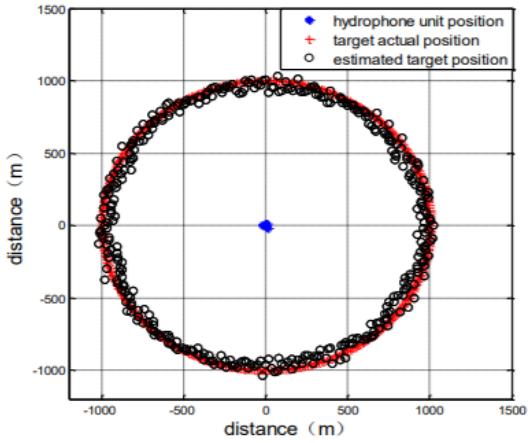
(a) Hydrophone positions



(b) Source localisation performance



(c) Hydrophone positions



(d) Source localisation performance

Figure 6-6: Simulated performance results for two different hydrophone configurations, for the task of source localisation. In (a) and (b), even though a third non-colinear hydrophone is used, its improper placement does not fully resolve ambiguities. In (c) and (d), a fourth hydrophone is used and the hydrophones are more uniformly distributed and therefore ambiguities are fully resolved. The simulated signal used was broadband with frequencies in the range 1 kHz to 10 kHz, the signal propagation speed was 1500 m/s and the SNR of the hydrophone signal readings was 10 dB (Wang et al. 2019).

underwater sound propagates in marine environments. Then, a variety of the most significant marine acoustic sources were presented, along with the specific characteristics of the signals that they emit. This information will be used in the next chapter to construct simulation of acoustic source localisation scenarios. Section 6.3 presented the basic principles of wavefield correlation using the two-hydrophone receiver model and it described how this model can be extended to more complex hydrophone configurations. These techniques will be fused with the PSO in the next chapter, to enhance its maximum range and convergence speed capabilities.

References

Au, W. W. (1993), 'The Sonar of Dolphins'.

Cummings, W. C. & Holliday, D. V. (1987), 'Sounds and source levels from bowhead whales off Pt. Barrow, Alaska', *The Journal of the Acoustical Society of America* **82**(3), 814–821.

Cummings, W. C. & Thompson, P. O. (1994), 'Characteristics and seasons of blue and finback whale sounds along the US west coast as recorded at SOSUS stations', *The Journal of the Acoustical Society of America* **95**(5), 2853.

Federation of American Scientists (1998), 'Sonar Propagation'.
URL: *https://fas.org/man/dod-101/navy/docs/es310/SNR_PROP/snr_prop.htm*

Fortmann, T., Bar-Shalom, Y. & Scheffe, M. (1983), 'Sonar tracking of multiple targets using joint probabilistic data association', *IEEE Journal of Oceanic Engineering* **8**(3), 173–184.

Gao, W., Wang, W., Zhu, H., Huang, G., Wu, D. & Du, Z. (2018), 'Robust Radiation Sources Localization Based on the Peak Suppressed Particle Filter for Mixed Multi-Modal Environments', *Sensors (Basel, Switzerland)* **18**(11), 3784.
URL: *https://pubmed.ncbi.nlm.nih.gov/30400670/*

Greene Jr, C. R. (1987), 'Characteristics of oil industry dredge and drilling sounds in the Beaufort Sea', *The Journal of the Acoustical Society of America* **82**(4), 1315–1324.

Hall, J. D., Gallagher, M. L., Brewer, K. D., Regos, P. R. & Isert, P. E. (1994), 'ARCO Alaska, Inc. 1993 Kuvlum Exploration Area Site Specific Monitoring Program. Final Report. Anchorage, AK: ARCO Alaska', *Inc. 218pp* .

Hao, B., Zhu, J., Li, Z., Xiao, S. & Tong, L. (2015), Passive Radar Source Localization Based on PSAAA Using Single Small Size Aircraft, *in* '2015 IEEE Globecom Workshops (GC Wkshps)', pp. 1–6.

Hildebrand, J. (2005), Impacts of Anthropogenic Sound.

Jiang, J., Liu, H., Duan, F., Wang, X., Fu, X., Li, C., Sun, Z. & Dong, X. (2019), 'Self-Contained High-SNR Underwater Acoustic Signal Acquisition Node and Synchronization Sampling Method for Multiple Distributed Nodes', *Sensors* **19**(21).
URL: *https://www.mdpi.com/1424-8220/19/21/4749*

Kundu, T. (2014), 'Acoustic source localization', *Ultrasonics* **54**(1), 25–38.
URL: *https://www.sciencedirect.com/science/article/pii/S0041624X13001819*

Li, P., Zhang, X. & Zhang, W. (2019), 'Direction of Arrival Estimation Using Two Hydrophones: Frequency Diversity Technique for Passive Sonar', *Sensors* **19**(9).
URL: *https://www.mdpi.com/1424-8220/19/9/2001*

Makowski, C. & Finkl, C. W. (2016), *History of Modern Seafloor Mapping*, Springer International Publishing, Cham, pp. 3–49.
URL: *https://doi.org/10.1007/978-3-319-25121-9_1*

Malgoezar, A., Snellen, M., Sijtsma, P. & Simons, D. (2016), Improving beamforming by optimization of acoustic array microphone positions.

Miller, P. J. & Tyack, P. L. (1998), 'A small towed beamforming array to identify vocalizing resident killer whales (Orcinus orca) concurrent with focal behavioral observations', *Deep Sea Research Part II: Topical Studies in Oceanography* **45**(7), 1389–1405.
URL: *https://www.sciencedirect.com/science/article/pii/S0967064598000289*

Munoz, D., Bouchereau, F., Vargas, C. & Enriquez, R. (2009), CHAPTER 2 - Signal Parameter Estimation for the Localization Problem, Academic Press, Oxford, pp. 23–65.
URL: *https://www.sciencedirect.com/science/article/pii/B9780123743534000089*

Pomareda, V., Magrans, R., Jiménez-Soto, J. M., Martínez, D., Tresánchez, M., Burgués, J., Palacín, J. & Marco, S. (2017), 'Chemical Source Localization Fusing Concentration Information in the Presence of Chemical Background Noise', *Sensors (Basel, Switzerland)* **17**(4), 904.
URL: *https://pubmed.ncbi.nlm.nih.gov/28425926/*

Richardson, W. J., Greene Jr, C. R., Malme, C. I. & Thomson, D. H. (2013), *Marine mammals and noise*, Academic press.

Rossing, T. D. & Fletcher, N. H. (2004), *Underwater Sound*, Springer New York, New York, NY, pp. 294–307.
URL: *https://doi.org/10.1007/978-1-4757-3822-3_13*

Rost, S. & Thomas, C. (2002), 'ARRAY SEISMOLOGY: METHODS AND APPLICATIONS', *Reviews of Geophysics* **40**(3), 2–27.
URL: *https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2000RG000100*

Stewart, J. L. & Westerfield, E. C. (1959), 'A Theory of Active Sonar Detection', *Proceedings of the IRE* **47**(5), 872–881.

Thompson, P. O., Cummings, W. C. & Ha, S. J. (1986), 'Sounds, source levels, and associated behavior of humpback whales, Southeast Alaska', *The Journal of the Acoustical Society of America* **80**(3), 735–740.

Triwahyanti, L., Cyndana, A., Sefnianti, Y., Sari, R. & Amron, A. (2018), 'Transmission Loss Estimation of Underwater Sound Based on the Noise Intensity Emmited by MV. Pengayoman IV in Tanjung Intan Cruise Line, Cilacap', *E3S Web of Conferences* **47**, 4011.

Urick, R. J. (1983), 'The noise background of the sea: ambient noise level', *Principles of Underwater Sound (ed. RJ Urick)* pp. 202–236.

Wang, Z., Li, L., Li, G. & Wang, Y. (2019), 'A research on blind area of random layout units for localization', *Journal of Physics: Conference Series* **1303**, 12064.
URL: *https://doi.org/10.1088/1742-6596/1303/1/012064*

Watkins, W. A. (1980), Acoustics and the behavior of sperm whales, *in* 'Animal sonar systems', Springer, pp. 283–290.

Wenz, G. M. (1962), 'Acoustic Ambient Noise in the Ocean: Spectra and Sources', *The Journal of the Acoustical Society of America* **34**(12), 1936–1956.
URL: *https://doi.org/10.1121/1.1909155*

Würsig, B., Clark, C., Montague, J. & Cowles, C. (1993), 'Behavior', *The Bowhead Whale* pp. 157–199.

Yack, T. M., Barlow, J., Calambokidis, J., Southall, B. & Coates, S. (2013), 'Passive acoustic monitoring using a towed hydrophone array results in identification of a previously unknown beaked whale habitat', *The Journal of the Acoustical Society of America* **134**(3), 2589–2595.
URL: *https://doi.org/10.1121/1.4816585*

Chapter 7

Wavefield Correlation-Enhanced Particle Swarm Optimisation

So far the focus of this thesis has been on the modification of PSO for the optimised control of robotic swarms. In the simulations of previous chapters, PSO operated by assigning a cost to each location based on the distance to the source. Clearly, this is not a suitable cost-assigning method for use in real-world robotic applications, since the position of the source is not known and the aim of the swarm is to identify it.

This chapter will address this gap by focusing on ways that the swarm can converge to a marine source, based on information that exists in the signals emitted by it. This will be achieved through further modifications of PSO. In contrast to previous chapters, the proposed modifications will concentrate on the way that personal and global best locations are selected, instead of the command outputs of the control algorithm. The chapter will focus on acoustic signals emitted by marine sources and therefore, the proposed modifications will aim to incorporate wavefield correlation capabilities into PSO, as discussed in Chapter 6.

7.1 Amplitude-Particle Swarm Optimisation (A-PSO)

The simplest way that the fitness of a location can be calculated in PSO for source localisation, is using the average intensity of the observed signal. As the signal propagates away from the source, it attenuates causing its intensity to drop. Therefore convergence to the source can be achieved by identifying the location where the signal intensity is maximised. The advantage of this method is that it can be used for a large variety of

signals, as long as they attenuate monotonically relative to the distance from the source. Examples of PSO in robotic swarms that employ this way of fitness assignment are in chemical source localisation using olfaction (Marques et al. 2006, Meng et al. 2011, Feng et al. 2019). This section will describe how this method could be implemented for acoustic signals.

Let f_A be the fitness function that needs to be maximised such that, at time $t_k = k\ \Delta t$ where k is the current timestep and Δt is the timestep size as defined in Chapters 3 to 5, the fitness of location $\mathbf{x}$ is given by

$$f_A(t_k, \mathbf{x}) = \frac{1}{T} \int_{-T}^{0} |s(t_k + t, \mathbf{x})|^2 dt, \tag{7.1}$$

where $s(t_k + t, \mathbf{x})$ is the observed signal at position $\mathbf{x}$ and T is its duration (i.e. window length). For the rest of this thesis, this type of PSO will be referred to as Amplitude-PSO (A-PSO), due to the use of signal intensity (amplitude) to assign fitness to a location.

7.1.1 Forgetting Function

As discussed in Chapter 2, a limitation of traditional PSO is that it can suffer from the *Outdated Memory Problem* (Blackwell 2007) in mutable/dynamic environments. As a reminder, in traditional PSO, each robot remembers its personal best location $\mathbf{y}$ until it discovers a new location of higher fitness. When dynamic fitness functions are used (i.e. moving source, mutable environment, noise etc.), the fitness of a location can change over time. The Outdated Memory Problem occurs when the fitness of a personal best location drops which should mean that it should be replaced with a new personal best location. Instead, the robot unaware of the change in fitness, still remembers the past fitness of the location, resulting in wrong decision and incapability of the swarm to converge properly to the source.

To address this problem, several PSO variants introduced a forgetting property that aims to slowly decrease the fitness of the current personal best location as remembered by the robot (Carlisle & Dozier 2000, Fernandez-Marquez & Arcos 2009), making it easier for it to be updated. Fusing a typical forgetting function (White 2001) with (7.1), the remembered fitness $\hat{f}^i[k]$ of the personal best location $\mathbf{y}^i[k]$ for robot i at timestep k is given by

$$\hat{f}^i[k] = \left\{ \begin{array}{ll} f_A(t_k, \mathbf{x}^i[k]), & \text{for } f_A(t_k, \mathbf{x}^i[k]) > \hat{f}^i[k-1] \\ \hat{f}^i[k-1] \times e^{-a}, & \text{otherwise} \end{array} \right\} \tag{7.2}$$

where a is a positive scalar. This describes both the cases where a new personal best location of higher fitness is found (i.e. $f_A(t_k, \mathbf{x}^i[k]) > \hat{f}^i[k-1]$) and where the old personal best location is maintained (i.e. forgetting function is applied). The overall personal best and global best location selection process is described by Algorithm 3, where $\mathbf{y_g}$ and $\hat{f}_g$ are the global best location and its fitness respectively and are assumed to be global variables known by all robots at all times.

Algorithm 3: Personal and global best location selection for A-PSO

```
foreach robot do
    robot.x ← robot.GetCurrentPosition();
    s[] ← robot.ReadSignal(); // Get sensor reading
    f_A ← mean(abs(s[])^2); // Assign fitness to selected intersection
    robot.UpdatePersonalBestLocation(f_A);
end
// Calculate global best location
f̂_g ← 0; // Reset global best location fitness
foreach robot do
    if robot.f̂ > f̂_g then // Update global best location
        f̂_g ← robot.f̂;
        y_g ← robot.y;
    end
end

Function UpdatePersonalBestLocation(self, fitness) :
    if fitness > self.f̂ then // Update personal best location
        self.f̂ ← fitness;
        self.y ← self.x;
    else // Apply forgetting function
        self.f̂ = self.f̂ × e^(−a);
    end
end
```

7.1.2 A-PSO Weaknesses

Even though A-PSO offers the advantage of being generally applicable to a large variety of source localisation problems, it is possible to identify several weaknesses that come with it:

1. As A-PSO relies on the intensity of the received signal, its performance is affected by the amount of noise in the signal. The use of a forgetting function allows A-PSO to achieve convergence eventually, even in the presence of noise, but its performance is expected to decrease as the SNR reduces. This in turn will limit

the maximum range of the algorithm and its ability to follow the gradient of the fitness function.

2. Each robot can only calculate the fitness of its current position and therefore, the exploration capabilities of the swarm depend on its span. This is referred to as the *Diversity Loss Problem* (Blackwell 2007). In parameter optimisation tasks, this is addressed by initialising the particle swarm around the area where the global fitness extremum is expected to exist. That said, in real-world source localisation applications, due to limited communication range, the swarm will most probably span over a small area relative to the large areas that will need to be explored. In this case, its exploration and convergence capabilities will be limited.

These weaknesses can be improved by incorporating wavefield correlation into the A-PSO algorithm, and it is this idea that will be explored in the rest of this chapter.

7.2 Wavefield Correlation PSO

Instead of measuring only the signal intensity with a single sensor it is possible to correlate the signals from a pair of sensors. In the presence of noise, the correlation of two signals can offer ways to address the negative effects of noise, theoretically achieving faster and over-time more consistent convergence towards the source, as described in Chapter 6. Additionally, the information collected from two sensors can be combined to calculate the direction to the source. In a robotic scenario, wavefield correlation can be used to calculate the angle of arrival (AOA) (Munoz et al. 2009) of a signal at a pair of wavefield sensors (e.g. two hydrophones) located on a robot, as shown in Figure 6-3. This section will offer a short summary of the types of information that can be extracted from this process, while a more detailed description is offered in Chapter 6.

Given two sensor readings $s_1(t)$ and $s_2(t)$ such that

$$\begin{aligned} s_1(t) &= s(t) + n_1(t), \\ s_2(t) &= s(t + \tau_{lag}) + n_2(t), \end{aligned} \tag{7.3}$$

where $s(t)$ is the received signal, $n_1(t)$ and $n_2(t)$ represent uncorrelated noise and τ_{lag} is the time difference between the signal being received by each hydrophone. As discussed in Chapter 6, using cross-correlation, the time difference τ_{lag}, the non-normalised correlation coefficient ρ and AOA α of the signal $s(t)$ can be found using

$$\tau_{lag} = \text{argmax}_\tau\{R(\tau)\}, \tag{7.4}$$

$$\rho = R(\tau_{lag}) = \max_{\tau}\{R(\tau)\} \tag{7.5}$$

and

$$\alpha = \cos^{-1}\left(\frac{\tau_{lag}\ c}{D}\right), \tag{7.6}$$

respectively, where τ is the lag, $R(\tau)$ is the cross-correlation of the two sensor readings, D is the sensor separation and c is the propagation speed of the signal.

The AOA of the signal is calculated with respect to the orientation of the robot. As discussed in Chapter 6, since only two sensors are used, there always exist two candidate directions of arrival as shown in Figure 6-4a, where they are referred to as directional rays. The directions of the directional rays associated with α and $-\alpha$ are given by

$$\mathbf{r}_{+}[k] = \begin{bmatrix}\cos(\alpha + o[k])\\ \sin(\alpha + o[k])\end{bmatrix} \qquad \mathbf{r}_{-}[k] = \begin{bmatrix}\cos(-\alpha + o[k])\\ \sin(-\alpha + o[k])\end{bmatrix}, \tag{7.7}$$

respectively, where $o[k]$ is the orientation of the robot at timestep k.

Additional ambiguities can be introduced due to the sensor separation D as shown in Figure 6-4b, when the signal is narrowband. To avoid this, the sensor separation can be limited to $D \leq \lambda_c/2$, where λ_c is the wavelength corresponding to the central frequency of the signal (Munoz et al. 2009). As the bandwidth of the signal increases, these types of ambiguities are generally reduced. The rest of this section will show how the wavefield information obtained through cross-correlation can be used to improve the weaknesses of A-PSO, discussed in Section 7.1.

7.2.1 Cross-Correlation Particle Swarm Optimisation (X-PSO)

The first algorithm that will be introduced in this chapter is Cross-Correlation PSO (X-PSO) and aims to address Weakness 1 of A-PSO. Weakness 1 describes how the range and convergence speed of A-PSO can be negatively affected by low SNRs. As described in Chapter 6, the SNR of the signal can be improved by cross-correlating the signals from multiple hydrophones. In this case, since two hydrophones are used, the intensity-based fitness used in A-PSO can be replaced with the non-normalised correlation coefficient ρ obtained using Equation (7.5), thereby reducing the negative effects of noise. This can be achieved by replacing f_A with f_X given by,

$$f_X(t_k, \mathbf{x}) = \max_{\tau}\left\{\int_{-T}^{0} s_1(t_k + t, \mathbf{x}) s_2(t_k + t + \tau, \mathbf{x}) dt\right\}, \tag{7.8}$$

Therefore, the fitness $\hat{f}^i[k]$ of the personal best location $\mathbf{y}^i[k]$ as remembered by robot i at timestep k is given by

$$\hat{f}^i[k] = \left\{ \begin{array}{ll} f_X(t_k, \mathbf{x}^i[k]), & \text{for } f_X(t_k, \mathbf{x}^i[k]) > \hat{f}^i[k-1] \\ \hat{f}^i[k-1] \times e^{-a}, & \text{otherwise} \end{array} \right\}. \tag{7.9}$$

Note that Equation (7.9) is identical to Equation (7.2) but $f_A(t_k, \mathbf{x}^i[k])$ is replaced with $f_X(t_k, \mathbf{x}^i[k])$. The overall personal and global best selection process is described by Algorithm 4.

Algorithm 4: Personal and global best location selection for X-PSO

```
foreach robot do
    robot.CalculateCorrelation();
    f_X ← robot.ρ; // Assign fitness to current location
    robot.UpdatePersonalBestLocation(f_X); // As in Algorithm 1
end
// Calculate global best location
f̂_g ← 0; // Reset global best location fitness
foreach robot do
    if robot.f̂ > f̂_g then // Update global best location
        f̂_g ← robot.f̂;
        y_g ← robot.y;
    end
end

Function CalculateCorrelation(self) :
    self.x ← self.GetCurrentPosition();
    s_1[], s_2[] ← self.ReadSignals(); // Get sensor readings
    R[] ← xcorr(s_1[],s_2[]); // Cross-correlate signals
    I_lag ← argmax(R[]);
    self.ρ ← R[I_lag]; // Calculate correlation coefficient
end
```

The advantage of X-PSO is that making use of the non-normalised correlation coefficient ρ from two different hydrophone readings can minimise the effects of noise and therefore, $f_X(t_k, \mathbf{x}^i[k])$ is likely smoother than $f_A(t_k, \mathbf{x}^i[k])$. This improves Weakness 1 of A-PSO and should make X-PSO more robust to noise. However, like A-PSO, X-PSO assigns fitness to the current location of each robot and therefore its exploration capabilities are limited by the span of the swarm. Therefore X-PSO does not address Weakness 2 of A-PSO. The next algorithm will aim to address this weakness.

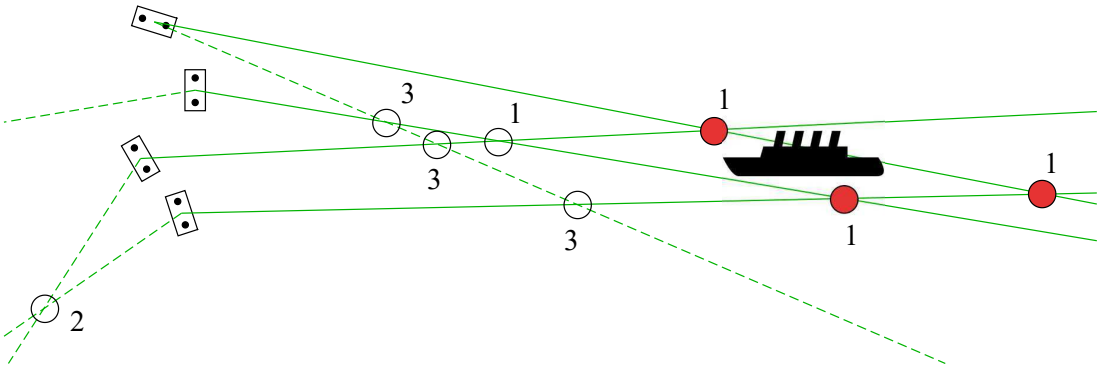


Figure 7-1: A swarm of robots (white rectangles) that uses B-PSO to estimate the location of an acoustic source (ship). Solid rays are source rays and dashed rays are ambiguous rays. The numbers next to each ray intersection (circles) represent the type of intersection. The red circles indicate the intersections that would likely be selected by at least one robot (i.e. furthest intersection from the robot). In this way, selected intersections are more likely to be intersections of source rays.

7.2.2 Bearing Particle Swarm Optimisation (B-PSO)

Assigning fitness to the current location of the robot is a fundamental characteristic of PSO, that leads to the problem of Diversity Loss (see Weakness 2 of A-PSO). Different PSO versions aim to address this by preventing the PSO particles from getting too close to each other. That said, when it comes to swarm robotic applications, this solution is not satisfying, since driving the robots too far apart from each other may result in communication loss between the robots. A possible alternative solution is therefore to modify PSO so that it assigns fitness to locations that are far away from the swarm. This can be achieved by using the directional rays of the robots (see Section 7.2) to estimate the location of the source, as shown in Figure 7-1. This approach is called Bearing-PSO (B-PSO).

In theory, at each timestep each robot could consider all intersections with all other robots of the swarm to maximise information gain, but this would require communication with all other robots, greatly increasing communication delays. Instead, at each timestep, each robot i receives the directional rays of only one other random robot j and combines it with its own to calculate ray intersections (Schneider & Eberly 2003). Combining each ray of robot i with each ray of robot j, can result from zero to four intersections at any timestep. The aim of robot i is to select the intersection that more closely approximates the location of the source.

Assuming that the SNR is significantly greater than 0 dB then on average at least one directional ray from each robot will point towards the source. Consider a single robot

rotating about its central axis; one ray will remain pointing towards the source whilst the other (the ambiguous $-\alpha$ ray in Figure 6-4a) will rotate throughout 360 degrees. Extending to a swarm of robots, the ambiguous (rotating) rays will likely lead to ray intersections that are close to the robots (distributed around the swarm), whilst the stable source rays are likely to lead to intersections that are distant (i.e., distributed around the source). Therefore, to increase the probability that an intersection of two unambiguous rays (i.e., those that point towards the source) is identified, robot i selects the *furthest* of its intersections from its current position.

A drawback of this process is that there is a risk that the selected intersections may be beyond the source or behind the swarm (especially for low SNR). Therefore, the most distal intersection (once selected) $\mathbf{p_s}^i$ is assigned a fitness f_B based on its distance from robot i given by

$$f_B(\mathbf{x}^i, \mathbf{p_s}^i) = \frac{1}{|\mathbf{x}^i - \mathbf{p_s}^i|}. \tag{7.10}$$

where $\mathbf{x}^i$ is the location of robot i. Thus, locations that are far away are progressively down weighted.

As with A-PSO and X-PSO, the selected intersection replaces the personal best location of robot i if its fitness is higher than the previous personal best location. Note that maximising the fitness function f_B does not necessarily imply convergence to the source. Thus a forgetting function is used to ensure that the personal best location is updated as robot i moves towards the source. In contrast to A-PSO and X-PSO, the forgetting function is fundamental for the correct operation of B-PSO, since it enables the continuous updating of the personal best locations. Therefore, the fitness $\hat{f}^i[k]$ of the personal best location $\mathbf{y}^i[k]$ of robot i at timestep k is given by (7.2) using $f_B(\mathbf{x}^i[k], \mathbf{p_s}^i[k])$ instead of $f_A(t_k, \mathbf{x}^i[k])$.

The outcome of the overall process described above is that personal best locations of high fitness are likely to be intersections of the first type which are assumed to be distributed around the source. Therefore, the centroid of the personal best locations best approximates the location of the source and is therefore used as the global best location $\mathbf{y_g}$,

$$\mathbf{y_g}[k] = \frac{\sum_{i=1}^{M} \mathbf{y}^i[k]}{M}, \tag{7.11}$$

where M is the total number of robots. The overall personal and global best location selection process for B-PSO is described by Algorithm 5.

The main advantage of B-PSO is that it selects personal best locations that are far

Algorithm 5: Personal and global best location selection for B-PSO

foreach *robot* **do**
  robot.CalculateCorrelation();
end
foreach *robot* **do** `// Select personal best locations`
  other ← robot.SelectRandomOtherRobot(); `// Uniformly random selection`
  $\mathbf{p}[] = []$;`// Initialise empty arrays of intersections`
  `// There can be up to 4 intersections`
  foreach *ray* ∈ *robot*.**r** **do**
    foreach *otherRay* ∈ *other*.**r** **do**
      p[].append(CalculateRayIntersections(robot.**x**, ray, other.**x**, otherRay));
    end
  end
  $d[] = |\text{robot}.\mathbf{x} - \mathbf{p}[]|$; `// Distances from robot to intersections`
  $\mathbf{p_s} \leftarrow \mathbf{p}[\text{argmax}(d[])]$; `// Select furthest intersection`
  $f_B \leftarrow 1/\max(d[])$; `// Assign fitness to selected intersection`
  robot.UpdatePersonalBestLocation($f_B, \mathbf{p_s}$);
end
`// Calculate global best location`
$\mathbf{y_g} \leftarrow [0, 0]$; `// Reset global best location`
foreach *robot* **do** `// Sum personal best locations`
  $\mathbf{y_g} \leftarrow \mathbf{y_g}$+robot.**y**;
end
$\mathbf{y_g} \leftarrow \mathbf{y_g}/M$; `// Calculate centroid`

Function *CalculateCorrelation(self)* **:**
  self.**x** ← self.GetCurrentPosition();
  $o \leftarrow$ self.GetCurrentOrientation();
  $s_1[], s_2[] \leftarrow$ self.ReadSignals(); `// Get sensor readings`
  $[R[], \tau[]] \leftarrow \text{xcorr}(s_1[], s_2[])$; `// Cross-correlate signals`
  $I_{lag} \leftarrow \text{argmax}(R[])$;
  self.$\rho \leftarrow R[I_{lag}]$; `// Calculate correlation coefficient`
  $\tau_{lag} \leftarrow \tau[I_{lag}]$; `// Calculate lag`
  $\alpha \leftarrow \text{acos}(\tau_{lag} * c/D)$;
  self.**r**[1] ← $[\cos(\alpha + o); \sin(\alpha + o)]$; `// Calculate directional rays`
  self.**r**[2] ← $[\cos(-\alpha + o); \sin(-\alpha + o)]$;
end

Function *UpdatePersonalBestLocation(self, fitness,* $\mathbf{p_s}$*)* **:**
  if *fitness* > *self.*$\hat{f}$ **then** `// Update personal best location`
    self.$\hat{f} \leftarrow$ fitness;
    self.**y** ← $\mathbf{p_s}$;
  else `// Apply forgetting function`
    self.$\hat{f}$ = self.$\hat{f} \times e^{-a}$;
  end
end

away from the swarm, thus fostering an exploration behaviour, and directly addressing Weakness 2 (i.e. its exploration capabilities are not limited by the span of the swarm). However, it is not clear at this point how efficiently B-PSO manages to avoid the negative effects of ambiguous rays (i.e. how often intersections of the second and third type are selected). If the selection process does not properly filter out such unhealthy intersections, it could lead to personal best locations that are located in the opposite direction from the source, reducing the overall performance of the algorithm.

7.2.3 Cross-Correlation-Bearing Particle Swarm Optimisation (XB-PSO)

Each of the weaknesses of A-PSO outlined in Section 7.1 are improved by either X-PSO or B-PSO. Nevertheless, both algorithms have their own weaknesses. Therefore, in an attempt to create an algorithm that has none of the weaknesses of B-PSO and X-PSO, both algorithms are combined to form a Cross-Correlation-Bearing PSO (XB-PSO) approach.

In XB-PSO both correlation coefficient ρ and directional ray information is communicated between robots. As outlined in Section 7.2.1, the source is likely closer to the robot with the highest correlation coefficient ρ. Therefore, among all intersections produced by the directional rays of robots i and j, each intersection $\mathbf{p}^{ij}$ can be validated using either of the two following conditions

$$\begin{aligned} &1) \quad |\mathbf{x}^i[k] - \mathbf{p}^{ij}[k]| < |\mathbf{x}^j[k] - \mathbf{p}^{ij}[k]| \quad \text{and} \quad \rho^i[k] > \rho^j[k] \\ &2) \quad |\mathbf{x}^i[k] - \mathbf{p}^{ij}[k]| > |\mathbf{x}^j[k] - \mathbf{p}^{ij}[k]| \quad \text{and} \quad \rho^i[k] < \rho^j[k] \end{aligned}, \tag{7.12}$$

where $\rho^i[k]$ and $\rho^j[k]$ are the non-normalised correlation coefficients of robots i and j respectively at timestep k. Intersections that do not satisfy either of these conditions are invalidated and cannot be selected as $\mathbf{p_s}^i$. From here, the process is identical to B-PSO, where the selected intersection $\mathbf{p_s}^i$ is assigned a fitness using Equation (7.10). This fitness is compared to the fitness $\hat{f}^i$ of the personal best location and if it is higher, the selected intersection becomes the new personal best location.

Furthermore, in XB-PSO, the selection of the global best location is performed in the same manner as B-PSO. The addition of the validation conditions of (7.12) aims to address the problem of B-PSO, of considering ray intersections that are in the opposite direction from the source, in an attempt to study how they affect the performance of B-PSO. The overall personal and global best location selection process is described by Algorithm 6.

Algorithm 6: Personal and global best location selection for XB-PSO

```
foreach robot do
    robot.CalculateCorrelation(); // As in Algorithm 3
end
foreach robot do // Select personal best locations
    other ← robot.SelectRandomOtherRobot(); // Uniformly random selection
    p[] = [];// Initialise empty arrays of intersections
    // There can be up to 4 intersections
    foreach ray ∈ robot.r do
        foreach otherRay ∈ other.r do
            p[].append(CalculateRayIntersections(robot.x, ray, other.x, otherRay));
        end
    end
    d[] = |robot.x − p[]|; // Distances from robot to intersections
    other_d[] = |other.x − p[]|; // Dist from other robot to intersections
    for q ← length(p[]) to 0 do // Remove non-valid intersections
        if robot.ρ > other.ρ and d[q] > other_d[q] then p[].remove(q);
        else if robot.ρ < other.ρ and d[q] < other_d[q] then p[].remove(q);
    end
    d[] = |robot.x − p[]|; // Distances from robot to intersections
    p_s ← p[argmax(d[])]; // Select furthest intersection
    f_B ← 1/max(d[]); // Assign fitness to selected intersection
    robot.UpdatePersonalBestLocation(f_B, p_s); // As in Algorithm 3
end
// Calculate global best location
y_g ← [0, 0]; // Reset global best location
foreach robot do // Sum personal best locations
    y_g ← y_g+robot.y;
end
y_g ← y_g/M; // Calculate centroid
```

It is clear that B-PSO and XB-PSO operate in a different way than traditional PSO. The capability of these algorithm to assign fitness to locations far away from the swarm can be beneficial in various ways. To identify this type of PSO variants from other more traditional variants, they will be referred to as triangulation PSO algorithms for the rest of this thesis.

7.3 Simulated Environment

In order to evaluate the performance of the proposed algorithms a set of simulations were performed using MATLAB. Due to the rise of autonomous marine systems, there has been recently an increasing interest in methods for the detection and localisation of unmanned underwater vehicles (UUV), primarily for defence purposes (Railey 2018, Railey et al. 2020, 2021, Open Cooperation for European mAritime awareNess 2021). Drawing from these scenarios, in these simulations a swarm of marine unmanned surface vehicles (USV) attempts to converge to an UUV that emits a continuous acoustic signal in moderate sea state. To approximate such a scenario, several parameter values were selected as shown in Table 7.1, based on a realistic scenario.

The simulated world is a 2D infinite plane. For simplicity, the presented simulations assume a stationary source and there is assumed to be no friction and no body inertia. Additionally, the signals are assumed to propagate at a constant speed . Each simulation was repeated 100 times, using Monte-Carlo sampling of the initial positions of robots, to obtain clear trends for the behaviour of each algorithm. The algorithms are assessed in terms of maximum range and rate of convergence towards the source.

7.3.1 Robots

The simulated USVs are equipped with two hydrophones located at the bow and stern of each USV. The swarm is initialised at random positions inside a circular area of radius d_0 around the origin, representing the area of initial deployment (e.g. a stationary ship). The velocity of each robot is directly controlled by the PSO controller defined in (3.11), allowing accurate control of the motion of the swarm and the maximum velocity V of the robots using (3.29). The robots are not forced to maintain a minimum separation from each other and collisions are ignored to test the algorithms in the extreme condition where the swarm spans over small area. In this way, it can be studied how the proposed algorithms address Weakness 2 of A-PSO, (i.e. since the exploration capabilities of A-PSO are limited by the spatial span of the swarm, it is desired to study whether the proposed algorithms can overcome this limitation).

Table 7.1: Selected parameter values to approximate a marine source localisation scenario

Parameter	**Value**	**Justification**
Timestep (Δt)	1 s	A typical high-level controller timestep size for robotic applications. Marine applications may also use larger values of Δt depending on the method of communication used (e.g. satellite communication).
Noise PSD (PSD_n)	60 dB re 1µPa^2/Hz	This is equivalent to a moderate sea state (Xerandy et al. 2015).
Source PSD (PSD_s)	120 dB re 1µPa^2/Hz at 1m	This is equivalent to the noise generated by a typical unmanned underwater vehicle (UUV) (Gebbie et al. 2012, Zimmerman et al. 2005).
Spatial step (Δx)	1000 m	Calculated using (7.17).
Source centre frequency (f_c)	1 kHz	The central frequency of a typical unmanned underwater vehicle (UUV) (Gebbie et al. 2012).
Maximum velocity (V)	2 m/s	Typical unmanned surface vehicle (USV) maximum speed range is 3 kn to 10 kn (Verfuss et al. 2019)).
Signal propagation speed (c)	1500 m/s	Speed of sound in water.
No. Robots (M)	10	A typical swarm size used in marine robotics (Griffiths Sànchez et al. 2018).
Starting radius (d_0) and convergence radius (d_c)	50 m	Resulting in large enough area to accommodate 10 robots of typical USV size.
Forgetting function scaling parameter (a)	1	Resulting in frequent updating of personal best locations.

7.3.2 Source

A single source, representing a UUV to be localised, is positioned at a distance d_s from the origin. Convergence to the source is considered to be achieved when the centre of mass of the swarm (CoM) reaches a distance d_c from the source at time t_c. The source is assumed to emit a continuous acoustic signal of constant power spectral density (PSD_s in units of dB re 1µPa^2/Hz at 1m) over a bandwidth B. This can be associated to the noise generated by a UUV motor by considering a narrow bandwidth B around the peak at frequency f_c of the power spectral density of the motor signal. For typical electrically propelled UUVs the power spectral density peaks at around 120 dB re 1µPa^2/Hz, occurring at frequency f_c from several hundred Hz to a few kHz (Gebbie et al. 2012, Zimmerman et al. 2005, Holmes et al. 2010). The simulated signal is therefore modelled using a filtered Gaussian process with central frequency f_c, bandwidth B and Q-factor given by

$$Q = \frac{f_c}{B} \tag{7.13}$$

The source level intensity I_s of the simulated signal is given by

$$I_s = 10^{\frac{\text{PSD}_s}{10}} \times 10^{-12} \times B \tag{7.14}$$

For the sake of generalisation, only the Q-factor of the signals will be provided in the results of the next section. This can allow the application of these results to different source localisation scenarios.

The signal is assumed to attenuate depending on the distance travelled, such that the intensity of the signal at distance r is given by

$$I = \frac{I_s}{r^2}. \tag{7.15}$$

Uncorrelated noise is added to the signals, modelled as filtered Additive White Gaussian Noise (AWGN) of constant power spectral density over the bandwidth B (PSD_n in units of dB re 1µPa^2/Hz) and average intensity N, approximating the ambient noise of a moderate sea state (Xerandy et al. 2015). The intensity N is related to PSD_n in the same way as I_s is related to PSD_s as described by (7.14). Therefore, the SNR at distance r from the source is given by

$$\text{SNR} = 10\log_{10}\left(\frac{I_s}{N}\right) - 10\log_{10}(r^2) \tag{7.16}$$

The SNR of the signals at the origin (i.e. the location of the initial deployment of the swarm), can be found using $r = d_s$ in (7.16) and is described using SNR_0. By varying the value of d_s, it is possible to vary the SNR experienced by the swarm at the beginning of the simulation (SNR_0). In this way, convergence of an algorithm from a larger distance d_s also implies more robustness to lower SNRs. Simulations were run for different values of d_s to test the ability of the algorithms to converge at different SNR_0 values.

7.3.3 Normalised Units and Parameter Values

In order to generalise the simulation results to a range of scenarios, normalised units of measurement are introduced. Normalised time is described in timesteps, such that each timestep has size Δt seconds as defined in Section 7.1. Normalised distance is given relative to a reference distance R, defined as the distance in metres at which $I = N$ (i.e. at $r = R$ m, SNR = 0 dB). Therefore, R is given by

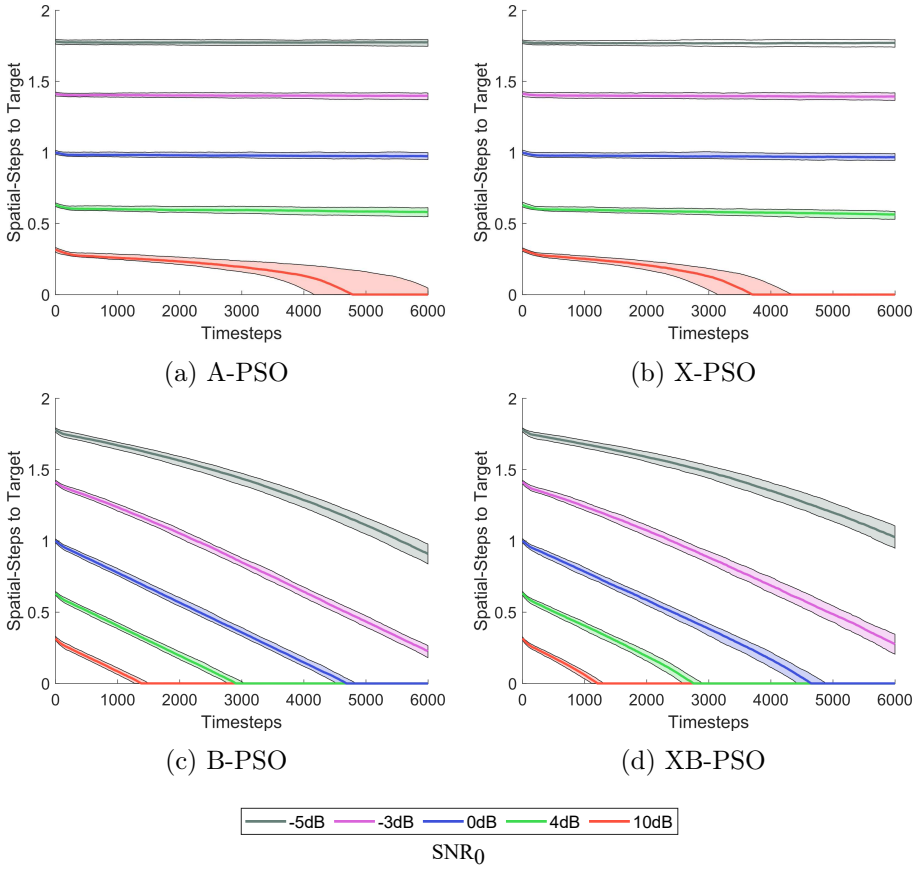
$$R = \sqrt{\frac{I_s}{N}} \tag{7.17}$$

Varying Δt allows the adaptation of the following results to scenarios with different controller loop delays (i.e. different communication rate between robots), while varying R enables the consideration of different signal intensities and noise levels.

7.4 Results

Two sets of simulations were performed to allow detailed comparison of the algorithms. In the first set, simulations were run to identify the ability of the algorithms to converge for different SNR_0 values. In the following simulation results, $Q = 1.5$ (i.e. one-octave band bandwidth). Also, the hydrophone separation for each robot is $D = \lambda_c/2$ and the time-bandwidth product of the cross-correlated signals is TBP = 100. The behaviour of the algorithms for different values of Q and D will be discussed further in the third set of simulations, in Section 7.4.1.

Figure 7-2 shows the median and 90% percentile ranges of the distance from the source of the CoM of the swarms as functions of time. Results are presented for each PSO variant at different SNR_0 values. The results show that A-PSO and X-PSO are only able to converge to the source when $\mathrm{SNR}_0 = 10$ dB, where A-PSO takes on average 4500 timesteps to reach convergence, while X-PSO takes on average 3400 timesteps. Therefore, X-PSO improves Weakness 1 of A-PSO. In contrast, both B-PSO and XB-



(a) A-PSO (b) X-PSO

(c) B-PSO (d) XB-PSO

Figure 7-2: Distances of CoM to source at different initial SNR values (SNR_0) for the tested algorithms. The solid lines represent the median distance from source over 100 simulations and the transparent areas the 90% percentile range.

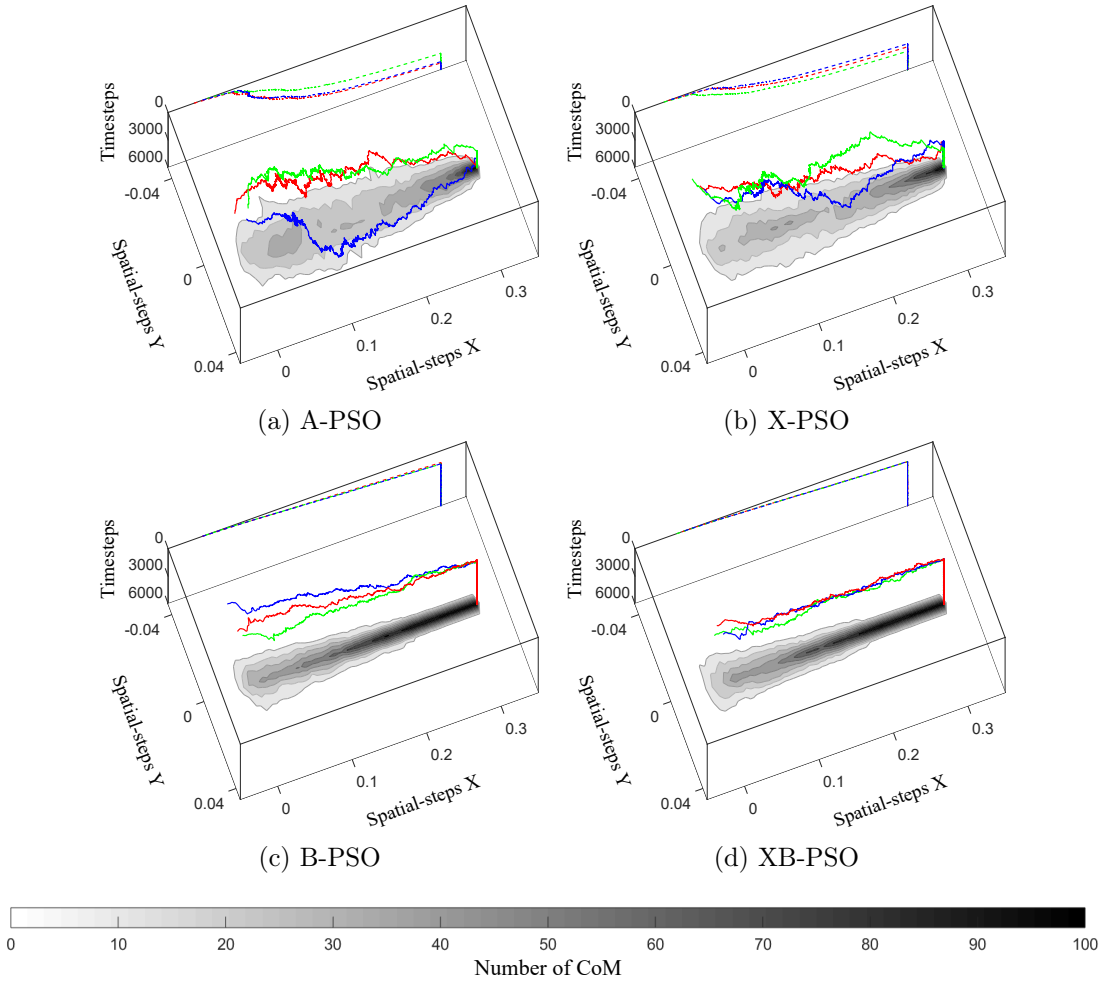


Figure 7-3: Graphical representation of the routes followed by the CoM of 100 swarms for each method. The image at the bottom plane of each graph shows contours of the number of swarms (CoM) that passed from different locations. The red, green and blue lines represent the routes followed by three randomly selected swarms with respect to time. The corresponding dashed lines represent the 2D projections of these routes on the Timesteps vs Spatial-steps vertical plane.

PSO are able to begin converging for all tested SNR_0 values, while exhibiting very similar convergence behaviours. B-PSO takes on average 1200, 2700 and 4400 timesteps to achieve convergence for $SNR_0 = 10$, 4 and 0 dB respectively. On the other hand, XB-PSO takes on average 1100, 2600 and 4400 timesteps to achieve convergence for the same SNR_0 values.

Since both B-PSO and XB-PSO employ personal best locations that are far away from the swarm, it is concluded that this results in a different behaviour that allows the algorithms to converge from further away. The next subsection will study how the spatial behaviour varies for each algorithm in an attempt to explain why B-PSO and XB-PSO have increased maximum range compared to the other two algorithms.

To better understand the features observed in Figure 7-2, the spatial routes followed by the simulated swarms are presented and analysed. Since all of the algorithms converge at $SNR_0 = 10\,\text{dB}$, only this case is studied in these results, presented in Figure 7-3. The results show that the two different ways of selecting personal best locations (near vs far away from the swarm), result in significantly different behaviours. A-PSO and X-PSO, which both select personal best locations near the swarm, appear to move in zig-zagging patterns, as shown by the behaviours of the randomly selected individual swarms. This can be interpreted as the algorithms finding it difficult to find their way to the source, (see Weakness 2 of A-PSO). On the other hand, B-PSO and XB-PSO, which can select personal best locations that are far away from the swarm result in fairly straight routes towards the source.

The contours of the graphs on the spatial axes (bottom of graphs) show similar results. A-PSO and X-PSO appear to spread more in the area and high probabilities ($> 50\%$) that a swarm will pass from a certain location only appear around 0.25 spatial-steps. On the other hand, B-PSO and XB-PSO exhibit less spreading and high probabilities that a swarm will pass from a certain location begin to appear early, at 0.1 spatial-steps. These results explain the results of Figure 7-2. At $10\,\text{dB}$ SNR_0, B-PSO and XB-PSO are much less affected by noise than A-PSO and X-PSO (i.e. their behaviour is more predictable and they appear to find it easier to move towards the source). Therefore, it makes sense that they appear to have such increased range in Figure 7-2 and it can be concluded that these algorithms address Weakness 2 of A-PSO.

7.4.1 Generalised Results

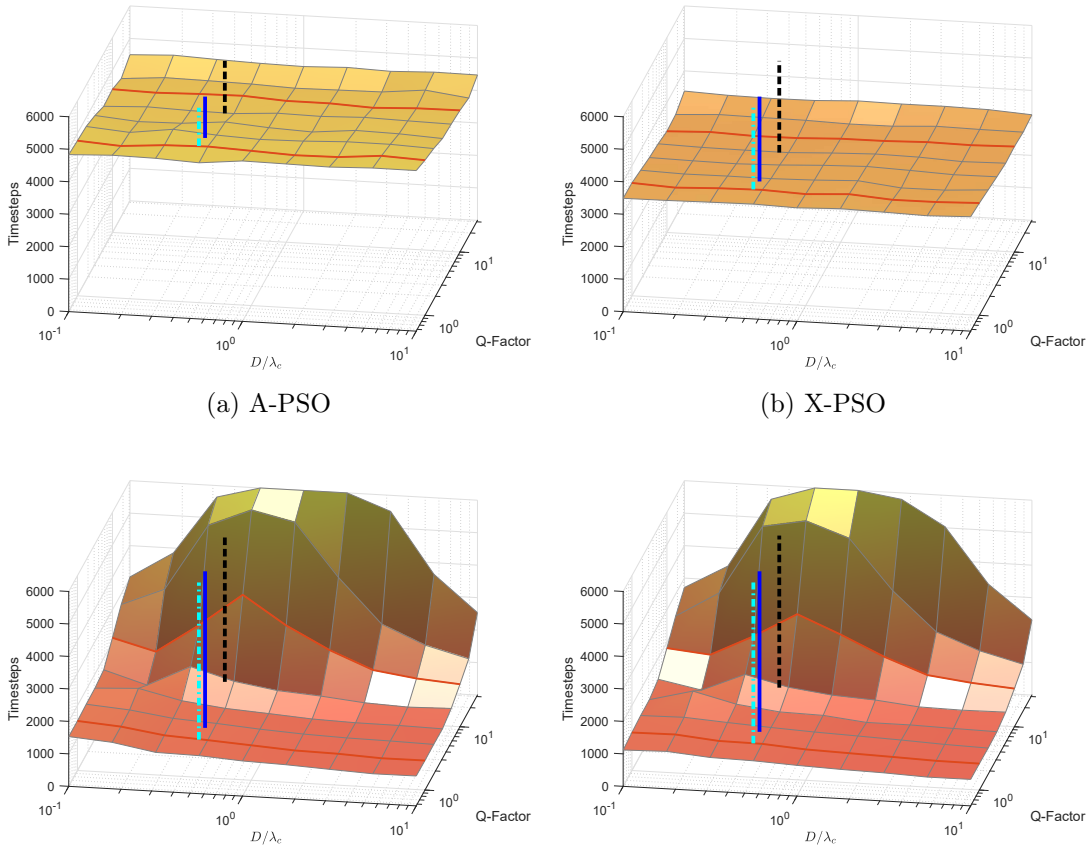
The results shown so far are presented in normalised time and distance in order to allow adjustment to different scenarios. However, the value of Q changes for different sources

(e.g. supertankers typically have much lower f_c than UUVs (Hildebrand 2005)) making it difficult to adjust the results. Additionally, the sensor separation D can also affect the performance of the algorithms. Lastly, changing the maximum velocity U of the robots can affected the convergence capabilities of the swarm, since robots will overshoot upon reaching their personal or global best location, resulting in better exploration of new locations. The rest of this section will present the results of simulations that were carried out using different values of Q, sensor separation D and maximum velocity U, for $\mathrm{SNR}_0 = 10\,\mathrm{dB}$. In this way, it is possible to predict how the results of Figure 7-2 and 7-3 can be altered to fit other scenarios.

Figure 7-4 shows the normalised median time to convergence over 100 repeats (to obtain an approximation of the behaviour of an average swarm), for all algorithms, for different values of Q and D, where D is normalised using the wavelength of the signal λ_c. The results show that the performance of B-PSO and XB-PSO decreases as Q increases. In contrast, A-PSO and X-PSO are not affected by Q changes. This is an expected result because as Q increases, the signal approaches a sinusoid (narrowband signal). This can result in ambiguities in the lag output of the cross-correlation function, as discussed in Section 7.2. Since only B-PSO and XB-PSO make use of the lag output, they are the only ones affected by this. Despite the decrease in performance, XB-PSO still appears to converge slightly faster than B-PSO for all Q values.

Furthermore, X-PSO is also unaffected by changes to the sensor separation D (A-PSO is also unaffected but this is because it only uses a single hydrophone). B-PSO and XB-PSO on the other hand exhibit poorer performance as the sensor separation approaches λ for large values of Q. This is expected because as the sensor separation increases beyond $\lambda/2$, it results in ambiguities in the lag calculation. Regardless, as the separation approaches 10λ, the convergence time is reduced again. This is interpreted to be the result of swarm intelligence. Even though more ambiguities are introduced, they are more evenly distributed around the robot. Therefore, since the robots constantly change orientation, the errors caused by ambiguities for some robots are averaged out by the correct predictions of others. Furthermore, it should be noted that the algorithms are unaffected by changes in sensor separation for small values of Q (i.e. broadband signals).

Figure 7-5 shows the normalised median time to convergence over 100 repeats (as in Figure 7-4) for all algorithms, for different values of U and Q. The maximum value of U used is $12\,\mathrm{m/s}$, corresponding to the maximum speed that a typical USV can achieve (Verfuss et al. 2019). In these simulations the sensor separation is D/λ_c as in the simulations of Figure 7-2. Both A-PSO and X-PSO appear to perform better as

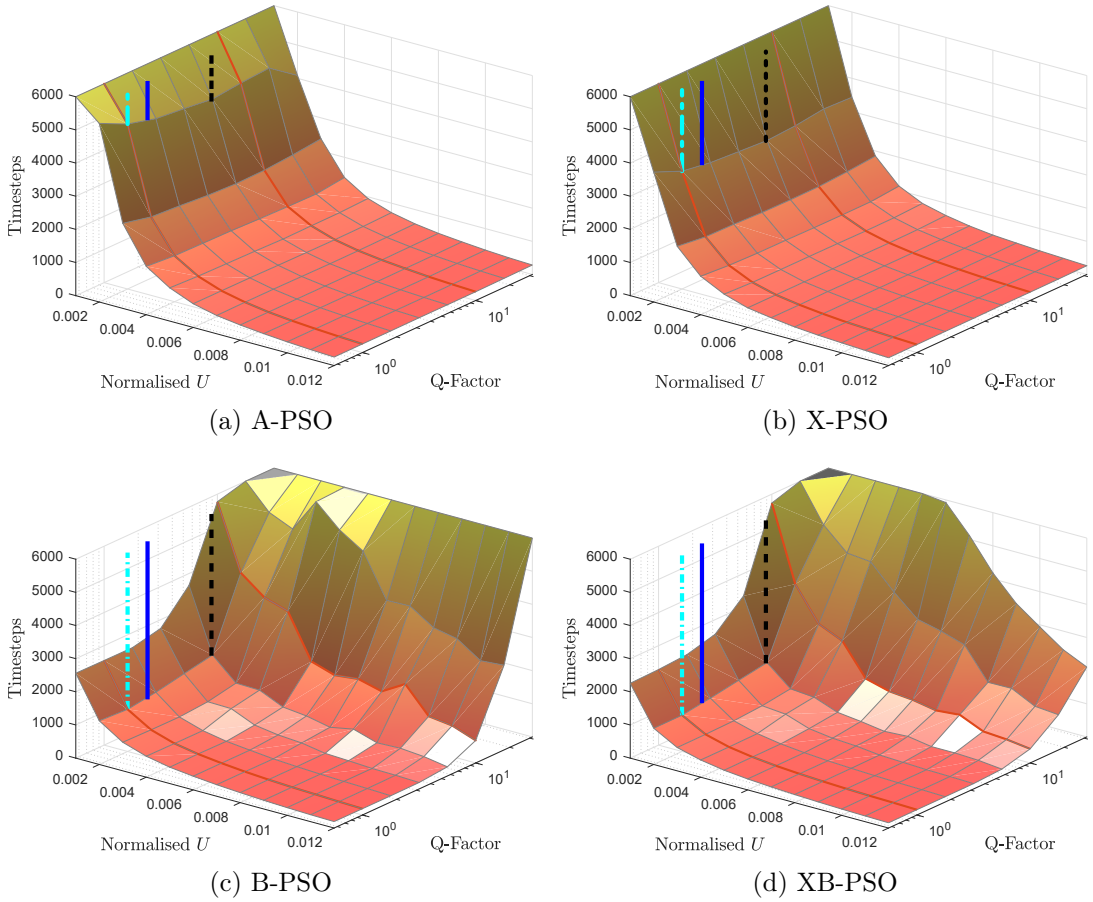


(a) A-PSO (b) X-PSO

(c) B-PSO (d) XB-PSO

Figure 7-4: Normalised time needed for the CoM of a swarm to reach convergence (distance d_c from the source) for each algorithm, for different values of Q and D/λ_c. Each point represents the median performance over 100 swarms. The red lines represent the locations where the Q-factor is 1 and 10. The vertical blue solid line represents the simulations of Figure 7-2. The vertical black dashed and cyan dashed-dotted lines represent the Q-factors of the sound generated by two electrically-propelled UUVs, REMUS-100 and Odyssey IIb respectively (Gebbie et al. 2012, Zimmerman et al. 2005, Holmes et al. 2010). Note that although a D/λ_c axis is included for A-PSO, it does not affect the position of the hydrophone, since only one is used for that algorithm, located in the middle of the robot.

U increases. These results are intuitive, since the time required to achieve convergence seems to be reduced in an inversely proportional manner to the increase of U, implying that the algorithms exhibit similar but faster behaviour to Figure 7-2 (i.e. the time to convergence is proportional to $\frac{d_s}{U}$). In agreement with Figure 7-4, both performances of A-PSO and X-PSO are independent of the value of Q. B-PSO and XB-PSO perform worse as Q increases, as suggested by Figure 7-4 and better as U increases. Furthermore,



(a) A-PSO (b) X-PSO

(c) B-PSO (d) XB-PSO

Figure 7-5: Normalised time needed for the CoM of a swarm to reach convergence (distance d_c from the source) for each algorithm, for different values of U and Q. Each point represents the median performance over 100 swarms. The red lines represent the locations where the Q-factor is 1 and 10. The vertical blue solid line represents the simulations of Figure 7-2. The vertical black dashed and cyan dashed-dotted lines represent the Q-factors of the sound generated by two electrically-propelled UUVs, REMUS-100 and Odyssey IIb respectively (Gebbie et al. 2012, Zimmerman et al. 2005, Holmes et al. 2010).

like A-PSO and X-PSO, their time to convergence also appears to reduce inversely proportionally to the increase of U. That said, a large difference in the performance of the two algorithms can be seen in the area where large values of both U and Q are used. Here, B-PSO is unable to achieve convergence no matter how much U increases. In contrast, XB-PSO's performance becomes better as U increases. This is the largest difference in performance between the two algorithms that is observed throughout all of the results of this section (i.e. in all other figures, XB-PSO performs better than B-PSO but the differences are small). This difference suggests that XB-PSO may offer

the capability to invalidate intersections that occur from ambiguities caused by high Q (i.e. as seen in Figure 6-4b). This is an interesting capability that could be explored in the future to improve XB-PSO further.

It can be therefore concluded from the presented results of this section that for most applications XB-PSO offers a good choice of control algorithm for robotic swarms, especially when the sensor separation can be selected or broadband signals are used. On the other hand, for specific applications where narrowband signals are used and the sensor separation is limited, X-PSO may provide a more robust alternative. As examples, Figures 7-4 and 7-5 include the cases of REMUS-100 and Odyssey IIb, two typical, electrically-propelled UUVs (vertical black dashed and cyan dashed-dotted lines respectively). It can be seen that the fast convergence and low SNR requirements of XB-PSO can make it an ideal choice for the detection of relatively silent vehicles that emit such low Q-factor signals, using robotic swarms.

7.5 Conclusions

This chapter discussed different ways that PSO can be used to identify the location of a source, by assigning fitness to locations using the signals emitted by the source. A-PSO, which can be already found in olfaction applications, uses the average intensity of the received signal to assign fitness to locations. In the presence of noise, this approach limits the maximum range and convergence speed of the algorithm as well as its exploration capabilities. This chapter introduced three new PSO variants (X-PSO, B-PSO and XB-PSO) that make use of correlation information that exist in acoustic wavefields, to improve these weaknesses of A-PSO. The results presented show that X-PSO is less sensitive to noise than A-PSO, resulting in faster convergence. B-PSO and XB-PSO are modified versions of PSO that make use of personal best locations far away from the swarm, greatly increasing their maximum range by more than 5 times and convergence speed by more than 4 times compared to A-PSO. XB-PSO is considered to be the best performing algorithm proposed in this chapter, for the presented application of acoustic source localisation.

References

Blackwell, T. (2007), *Particle Swarm Optimization in Dynamic Environments*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 29–49.
URL: *https://doi.org/10.1007/978-3-540-49774-5_2*

Carlisle, A. & Dozier, G. (2000), 'Adapting Particle Swarm Optimization to Dynamic Environments', *Proc of Int Conf on Artificial Intelligence* .

Feng, Q., Zhang, C., Lu, J., Cai, H., Chen, Z., Yang, Y., Li, F. & Li, X. (2019), 'Source localization in dynamic indoor environments with natural ventilation: An experimental study of a particle swarm optimization-based multi-robot olfaction method', *Building and Environment* **161**, 106228.
URL: *https://www.sciencedirect.com/science/article/pii/S036013231930438X*

Fernandez-Marquez, J. L. & Arcos, J. L. (2009), An Evaporation Mechanism for Dynamic and Noisy Multimodal Optimization, *in* 'Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation', GECCO '09, Association for Computing Machinery, New York, NY, USA, pp. 17–24.
URL: *https://doi.org/10.1145/1569901.1569905*

Gebbie, J., Siderius, M. & Allen, J. S. r. (2012), 'Aspect-dependent radiated noise analysis of an underway autonomous underwater vehicle.', *The Journal of the Acoustical Society of America* **132**(5), EL351–7.

Griffiths Sànchez, N. D., Vargas, P. A. & Couceiro, M. S. (2018), A Darwinian Swarm Robotics Strategy Applied to Underwater Exploration, *in* '2018 IEEE Congress on Evolutionary Computation (CEC)', pp. 1–6.

Hildebrand, J. (2005), Impacts of Anthropogenic Sound.

Holmes, J. D., Carey, W. M. & Lynch, J. F. (2010), 'An overview of unmanned underwater vehicle noise in the low to mid frequencies bands', *Proceedings of Meetings on Acoustics* **9**(1), 65007.
URL: *https://asa.scitation.org/doi/abs/10.1121/1.3492795*

Marques, L., Nunes, U. & de Almeida, A. T. (2006), 'Particle swarm-based olfactory guided search', *Autonomous Robots* **20**(3), 277–287.
URL: *https://doi.org/10.1007/s10514-006-7567-0*

Meng, Q.-H., Yang, W.-X., Wang, Y. & Zeng, M. (2011), 'Collective Odor Source Estimation and Search in Time-Variant Airflow Environments Using Mobile Robots',

Sensors **11**(11), 10415–10443.
URL: *https://www.mdpi.com/1424-8220/11/11/10415*

Munoz, D., Bouchereau, F., Vargas, C. & Enriquez, R. (2009), CHAPTER 2 - Signal Parameter Estimation for the Localization Problem, Academic Press, Oxford, pp. 23–65.
URL: *https://www.sciencedirect.com/science/article/pii/B9780123743534000089*

Open Cooperation for European mAritime awareNess (2021), 'Ocean2020 Mediterranean Sea Trials'.
URL: *https://ocean2020.eu/mediterranean-sea/about-the-aim/*

Railey, K. (2018), 'Demonstration of passive acoustic detection and tracking of unmanned underwater vehicles'.

Railey, K., DiBiaso, D. & Schmidt, H. (2020), 'An acoustic remote sensing method for high-precision propeller rotation and speed estimation of unmanned underwater vehicles', *The Journal of the Acoustical Society of America* **148**(6), 3942–3950.
URL: *https://doi.org/10.1121/10.0002954*

Railey, K., Dibiaso, D. & Schmidt, H. (2021), 'Passive acoustic detection and tracking of an unmanned underwater vehicle from motor noise', *The Journal of the Acoustical Society of America* **149**(4), A34–A35.
URL: *https://doi.org/10.1121/10.0004444*

Schneider, P. J. & Eberly, D. H. (2003), CHAPTER 7 - INTERSECTION IN 2D, *in* P. J. SCHNEIDER & D. H. EBERLY, eds, 'Geometric Tools for Computer Graphics', The Morgan Kaufmann Series in Computer Graphics, Morgan Kaufmann, San Francisco, pp. 241–284.
URL: *https://www.sciencedirect.com/science/article/pii/B9781558605947500102*

Verfuss, U. K., Aniceto, A. S., Harris, D. V., Gillespie, D., Fielding, S., Jiménez, G., Johnston, P., Sinclair, R. R., Sivertsen, A., Solbø, S. A., Storvold, R., Biuw, M. & Wyatt, R. (2019), 'A review of unmanned vehicles for the detection and monitoring of marine fauna', *Marine Pollution Bulletin* **140**, 17–29.
URL: *https://www.sciencedirect.com/science/article/pii/S0025326X19300098*

White, K. G. (2001), 'Forgetting functions', *Animal Learning & Behavior* **29**(3), 193–207.
URL: *https://doi.org/10.3758/BF03192887*

Xerandy, X., Znati, T. & K, L. (2015), 'Cost-Effective, Cognitive Undersea Network

for Timely and Reliable Near-Field Tsunami Warning', *International Journal of Advanced Computer Science and Applications* **6**.

Zimmerman, R., D'Spain, G. L. & Chadwell, C. D. (2005), 'Decreasing the radiated acoustic and vibration noise of a mid-size AUV', *IEEE Journal of Oceanic Engineering* **30**(1), 179–187.

Chapter 8

Generalised Triangulation PSO and Source Entrapment/Escorting

So far this thesis has addressed several problems and limitations that prevented PSO from being used for the control of robotic swarms, as discussed in Chapter 2. Furthermore, Chapter 7 has addressed several weaknesses that were inherent to PSO due to the way that personal best locations are selected. This was achieved through the introduction of triangulation PSO algorithms that enable the swarm to consider personal best locations that are far away from the swarm. Nevertheless, there remains one limitation that prevents PSO from being used in source localisation applications. As discussed in Chapter 2, it is impossible for the swarm to know the location of a source before a robot has passed directly over it, which in turn prevents the swarm from interacting with the source (e.g. avoid it or maintain a constant distance from it).

Triangulation PSO algorithms may offer the ability to overcome this limitation. Instead of "remembering" the fitness/cost of a location, these algorithms estimate the location of the source using directional ray intersections, allowing the swarm to predict the source's location from far away and therefore interact with it. Furthermore, they make use of a forgetting function (White 2001), that forces them to consider new personal best locations, as time passes. This enables the robots to discard outdated information and, in Chapter 7 it was shown how this can be used to deal with a noisy fitness function (e.g. noise in the hydrophone readings) for a stationary source. Beyond this, the use of a forgetting function may allow triangulation PSO to also deal with dynamic fitness functions due to a moving source or mutable environments.

This chapter will explore how a triangulation PSO algorithm can be implemented, for

use in non-wavefield sources (i.e. using only signal intensity information). In marine applications, this can enable the localisation of heat sources, regions of high salinity, regions of high concentration of phytoplankton etc. Afterwards, the algorithm will be tested in MATLAB simulations, to confirm that it does indeed permit localisation of a moving source and interaction with it.

8.1 Generalised Triangulation PSO

The main requirement for the implementation of a triangulation PSO algorithm is for the robots to be able to calculate directional rays that point towards the source. It is impossible for a single robot to calculate directional rays on its own using only signal intensity information. Therefore, to calculate directional rays, the information from multiple sensors (i.e. multiple robots) needs to be combined. The rest of this section will show how this can be achieved for a 2D environment. A 3D approach can be also achieved using the proposed method, with minor modifications.

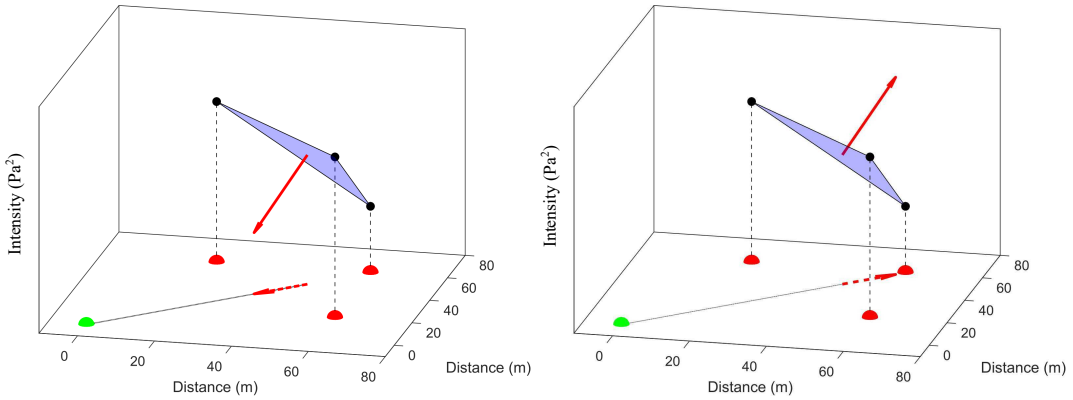
Since the robots make use of signal intensity information, the signal intensity at location $\mathbf{x}$ at time $t_k = k\ \Delta t$ can be calculated using

$$I(t_k, \mathbf{x}) = \frac{1}{T} \int_{-T}^{0} |s(t_k + t, \mathbf{x})|^2 dt, \tag{8.1}$$

where k is the current timestep, $s(t_k + t, \mathbf{x})$ is the observed signal at position $\mathbf{x}$ and T is its duration. Note that Equation (8.1) is equivalent to the fitness function used by A-PSO in Chapter 7. In this case though, the signal intensity will not be used for the selection of personal best locations and it will instead be used for the calculation of directional rays.

Each independent signal intensity measurement allows the recovery of one directional constraint. To uniquely identify the direction to the source in an n-dimensional space, one needs $n+1$ independent signal intensity measurements, where the $n+1^{th}$ is typically used to resolve ambiguities (Zhao & Guibas 2004). Therefore, to calculate a directional ray in a 2-dimensional environment, the signal intensity information from three different robots at the same timestep needs to be combined. This can allow the robots to estimate the gradient ∇I at the CoM of the three robots and use it to produce a directional ray.

Due to possible curvature of I (i.e. curvature of the wavefront of the signal), the estimation of ∇I is more accurate the closer the robots are to each other. Therefore, at each timestep, robot i receives the current position and signal intensity of its two closest robots j and h. The current position and signal intensity of each robot are



(a) (b)

Figure 8-1: Schematics that explain the operation of Generalised Triangulation PSO. The red spheres represent the locations of robots and the green sphere represents the location of the source. The height of each robot's position-intensity (black spheres) represents the average intensity of the sensor readings of the corresponding robot. The blue triangle is the position-intensity plane and the red arrow represents its normal. The dashed red arrow is the projection of the normal on the xy-plane and it is collinear with the source. In (a), the z-component of the normal is negative and therefore its projection points towards the source. In (b), the z-component of the normal is positive and therefore its projection points away from the source.

combined into a single vector, called the position-intensity point of each robot, such that the position-intensity points $\mathbf{q}^i[k]$, $\mathbf{q}^j[k]$ and $\mathbf{q}^h[k]$ of robots i, j and h at timestep k are given by

$$\mathbf{q}^i[k] = \begin{bmatrix} x_1^i[k] \\ x_2^i[k] \\ I(\mathbf{x}^i[k]) \end{bmatrix}, \quad \mathbf{q}^j[k] = \begin{bmatrix} x_1^j[k] \\ x_2^j[k] \\ I(\mathbf{x}^j[k]) \end{bmatrix} \quad \text{and} \quad \mathbf{q}^h[k] = \begin{bmatrix} x_1^h[k] \\ x_2^h[k] \\ I(\mathbf{x}^h[k]) \end{bmatrix}, \tag{8.2}$$

where $x_1^i[k]$ and $x_2^i[k]$ are the first and second components of the position $\mathbf{x}^i[k]$ of robot i at timestep k, and $I(\mathbf{x}^i[k])$ is the intensity of its sensor readings at $\mathbf{x}^i[k]$. For the sake of visualisation, in Figure 8-1, each black point describes the position-intensity point of each corresponding robot (red spheres).

The position-intensity points of robots i, j and h can now be combined to identify a plane, called the position-intensity plane, that passes through all three position-intensity points. The position-intensity plane is represented in Figure 8-1 by the blue triangle. Furthermore, let $\mathbf{m}^{i,j,h}$ be the midpoint of the positions $\mathbf{x}^i[k]$, $\mathbf{x}^j[k]$ and $\mathbf{x}^h[k]$, given by

$$\mathbf{m}^{i,j,h}[k] = \frac{\mathbf{x}^i[k] + \mathbf{x}^j[k] + \mathbf{x}^h[k]}{3}, \tag{8.3}$$

As the distance between the robots becomes smaller, the gradient of the position-intensity plane approaches $\nabla I(\mathbf{m}^{i,j,h})$.

The position-intensity plane can be used to identify the direction to the source. The normal $\mathbf{n}^{i,j,h}$ to the position-intensity plane at timestep k, represented by the red solid arrow in Figure 8-1, is given by

$$\mathbf{n}^{i,j,h}[k] = (\mathbf{q}^i[k] - \mathbf{q}^j[k]) \times (\mathbf{q}^i[k] - \mathbf{q}^h[k]), \tag{8.4}$$

where $\times$ is the cross-product. Theoretically, if I is convex-shaped and if the robots are close to each other, the projection of $\mathbf{n}^{i,j,h}$ on the xy-plane (i.e. the red dashed arrow on the horizontal plane in Figure 8-1) is parallel to the line formed by $\mathbf{m}^{i,j,h}$ and the source. Therefore, a directional ray can start from $\mathbf{m}^{i,j,h}$ and be parallel to the projection of $\mathbf{n}^{i,j,h}$ on the xy-plane.

Finally, depending on the positions of the three robots, the normal $\mathbf{n}^{i,j,h}$ can face either towards the source or away from it. These two cases are individually showcased by Figure 8-1a and Figure 8-1b respectively. The sign of the z-component of the normal can be used to identify which is the case. In other words, if $\mathbf{n}^{i,j,h}$ points downwards (i.e. its z-component $\mathbf{n}_z^{i,j,h} < 0$), its projection will point towards the source. Therefore, in this case, the directional ray will start from $\mathbf{m}^{i,j,h}$ and it will have the same direction as the projection of $\mathbf{n}^{i,j,h}$. Conversely, if $\mathbf{n}^{i,j,h}$ points upwards (i.e. its z-component $\mathbf{n}_z^{i,j,h} > 0$), its projection will point away from the source. Therefore, in this case, the directional ray will start from $\mathbf{m}^{i,j,h}$ and it will have the opposite direction to the projection of the normal. Note that if I is a cost function, the direction of the ray, as defined by the sign of $\mathbf{n}_z^{i,j,h}$ will be inverted.

Each robot can calculate a single directional ray, using the process described above. Therefore, the directional rays from two different robots can now be combined to calculate personal best locations for each robot, in a similar manner to B-PSO, as described in Chapter 7. Note that there is always a chance that robots i, j and h will share the same directional ray. Therefore, when robot i selects a random robot to compare directional rays, robots j and h should be excluded from the selection, in order to ensure that a single intersection can be derived from the comparison of the directional rays. Furthermore, since each robot only has a single directional ray, when two robots compare directional rays with each other, a maximum of only one intersection can occur (in contrast to B-PSO and XB-PSO where there could be up to four intersections). Therefore, that single intersection is directly selected as a candidate new personal best location. Like B-PSO and XB-PSO, a fitness f_B is applied on the selected intersection

$\mathbf{p_s}^i$ based on its distance from robot i such that

$$f_B(\mathbf{x}^i, \mathbf{p_s}^i) = \frac{1}{|\mathbf{x}^i - \mathbf{p_s}^i|}. \tag{8.5}$$

Note that Equation (8.5) is equivalent to the fitness function used by B-PSO and XB-PSO in Chapter 7. The selected intersection will replace the personal best location $\mathbf{y}^i$ of robot i if its fitness is higher than the fitness $\hat{f}^i$ of $\mathbf{y}^i$. Lastly, like B-PSO and XB-PSO, a forgetting function is applied to the fitness of $\mathbf{y}^i$. Therefore, the fitness $\hat{f}^i[k]$ of the personal best location $\mathbf{y}^i[k]$ of robot i at timestep k is given by

$$\hat{f}^i[k] = \left\{ \begin{array}{ll} f_B(t_k, \mathbf{x}^i[k]), & \text{for } f_B(t_k, \mathbf{x}^i[k]) > \hat{f}^i[k-1] \\ \hat{f}^i[k-1] \times e^{-a}, & \text{otherwise} \end{array} \right\} \tag{8.6}$$

The global best location is calculated in the same way as in B-PSO and XB-PSO, using the centroid of the personal best locations of all robots. The overall personal and global best location selection process is described by Algorithm 7. The resulting algorithm, is capable of convergence towards the source, in a similar manner to B-PSO and XB-PSO (see Chapter 7), but it operates using signal intensity measurements (or any other type of positional information). Despite their similarities, there can be identified two limitations in the Generalised Triangulation PSO that do not exist in B-PSO and XB-PSO:

1. The calculation of the directional rays in Generalised Triangulation PSO is more accurate the closer the robots are to each other. This limitation is less important when the robots are far away from the source (i.e. far-field, when the wavefront of the signal is flat), but it becomes more important as the robots get closer to the source (i.e. near-field). This is because, when close to the source, due to the curvature of the wavefront of the signal, it becomes more difficult to approximate the gradient ∇I using the gradient of the position-intensity plane.

2. Additionally, the rule that the calculation of the directional rays is more accurate the closer the robots are to each other, holds in an idealised scenario where there is no noise. In the presence of uncorrelated noise in the signal readings, the gradient of the position-intensity plane may vary randomly. When the robots are directly above each other (i.e. no collision), the difference in signal intensity is minimised and therefore the gradient of the position-intensity plane is more affected by the noise in the signal readings. Therefore, to prevent the robots from getting too close to each other, Generalised Triangulation PSO needs to be combined with

Algorithm 7: Personal and global best location selection for Generalised Triangulation PSO

foreach *robot* **do**
  robot.GetIntensityInfo();
end
foreach *robot* **do**
  [close_robot_1,close_robot_2] ← robot.SelectTwoClosestRobots();
  robot.$\mathbf{m} \leftarrow (\text{robot.}\mathbf{x} + \text{close\_robot\_1.}\mathbf{x} + \text{close\_robot\_2.}\mathbf{x})/3$;
  $\mathbf{n} \leftarrow \text{cross}((\text{robot.}\mathbf{q} - \text{close\_robot\_1.}\mathbf{q}), (\text{robot.}\mathbf{q} - \text{close\_robot\_2.}\mathbf{q}))$;
  robot.$\mathbf{r} \leftarrow -\text{sgn}(\mathbf{n}[3]) * \mathbf{n}[1:2]$;
  `// Save closest robots for use in next loop`
  robot.cr1 = close_robot_1; robot.cr2 = close_robot_2;
end
foreach *robot* **do** `// Select personal best locations`
  other_robot ← robot.SelectRandomOtherRobotExcept(robot.cr1,robot.cr2);
  `// There can be up to 1 intersection`
  $\mathbf{p_s} \leftarrow$ CalcRayIntersections(robot.$\mathbf{m}$, robot.$\mathbf{r}$, other_robot.$\mathbf{m}$, other_robot.$\mathbf{r}$);
  robot.UpdatePersonalBestLocation($\mathbf{p_s}$);
end
`// Calculate global best location`
$\mathbf{y_g} \leftarrow [0, 0]$; `// Reset global best location`
foreach *robot* **do** `// Sum personal best locations`
  $\mathbf{y_g} \leftarrow \mathbf{y_g}$+robot.$\mathbf{y}$;
end
$\mathbf{y_g} \leftarrow \mathbf{y_g}/M$; `// Calculate centroid`

Function *GetIntensityInfo(self)* **:**
  self.$\mathbf{x} \leftarrow$ self.GetCurrentPosition();
  $s[] \leftarrow$ self.ReadSignal(); `// Get hydrophone reading`
  $I \leftarrow \text{mean}(\text{abs}(s[])^2)$;
  self.$\mathbf{q}$ = [self.$\mathbf{x}$[1], self.$\mathbf{x}$[2], I]; `// Assemble Position-Intensity point`
end
Function *UpdatePersonalBestLocation(self,* $\mathbf{p_s}$*)* **:**
  $d = |\text{self.}\mathbf{x} - \mathbf{p_s}|$; `// Distance from robot to intersection`
  $f_B \leftarrow 1/d$; `// Assign fitness to selected intersection`
  if $f_B > \textit{self.}\hat{f}$ **then** `// Update personal best location`
    self.$\hat{f} \leftarrow f_B$;
    self.$\mathbf{y} \leftarrow \mathbf{p_s}$;
  else `// Apply forgetting function`
    self.$\hat{f} =$ self.$\hat{f} \times e^{-a}$;
  end
end

aggregation.

Therefore, in real-world scenarios, there exists a trade-off between how close the robots need to be to each other, to avoid the negative effects of the curvature of the signal wavefront, and how far away they have to be from each other to avoid the negative effects of noise. In the following subsection, the algorithm will be tested in 2D MATLAB simulations to test its convergence capabilities, which will be compared to the convergence capabilities of XB-PSO.

8.1.1 Comparison with XB-PSO

To test the ability of Generalised Triangulation PSO to converge to a source, simulations similar these of Chapter 7 were used. The algorithm's performance is compared to the convergence performance of XB-PSO. Since Generalised Triangulation PSO requires the inclusion of aggregation to operate properly, both algorithms were merged with the non-omnidirectional PSO controller, described in Chapter 5, to include aggregation and allow equal comparison.

Since the non-omnidirectional PSO controller deals with the control of the motion of the robots, while triangulation PSO algorithms deal with the calculation of the personal and global best locations, the two can be easily combined. In other words, in the following algorithms, the personal best location $\tilde{\mathbf{y}}$ and global best location $\tilde{\mathbf{y_g}}$ obtained using Generalised Triangulation PSO and XB-PSO will be used to form the first and second accelerating terms of the non-omnidirectional PSO velocity update equation as shown below

$$\tilde{\mathbf{u}}[k+1] = \boldsymbol{\omega} \circ \tilde{\mathbf{u}}[k] + \mathbf{c_1} \circ \mathbf{r_1} \circ \text{sgn}(\tilde{\mathbf{y}}[k]) + \mathbf{c_2} \circ \mathbf{r_2} \circ \text{sgn}(\tilde{\mathbf{y_g}}[k]) - \mathbf{c_3} \circ \mathbf{r_3} + \mathbf{c_4} \circ \mathbf{r_4} + \mathbf{c_5} \circ \mathbf{r_5}, \quad (8.7)$$

where $\tilde{\mathbf{u}}[k]$ is the relative velocity vector of the robot at timestep k, $\tilde{\mathbf{y}}$ and $\tilde{\mathbf{y_g}}$ are the positions of the personal best location and global best location relative to the robot and

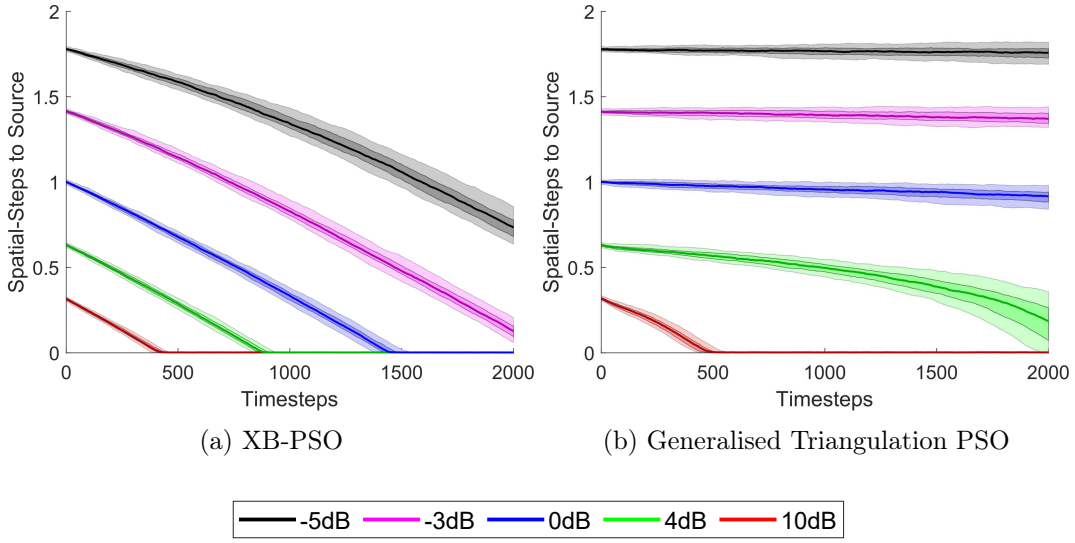
$$\mathbf{c_1} = \begin{bmatrix} 0 \\ c_1 \end{bmatrix}, \quad \mathbf{c_2} = \begin{bmatrix} 0 \\ c_2 \end{bmatrix}, \quad \mathbf{c_3} = \begin{bmatrix} 0 \\ c_3 \end{bmatrix}, \quad \mathbf{c_4} = \begin{bmatrix} 0 \\ c_4 \end{bmatrix}, \quad \mathbf{c_5} = \begin{bmatrix} c_5 \\ 0 \end{bmatrix}$$

In a similar manner to Chapter 7, the simulations are calibrated to approximate a marine robotic scenario. Furthermore, the non-omnidirectional PSO controller makes use of the same parameters (α and β) as in Chapter 5. The selected parameter values with justifications can be seen in Table 8.1.

The algorithm performance shown in Figure 8-2b, is the result of simulations carried

Table 8.1: Selected parameter values to approximate a marine source localisation scenario

Parameter	**Value**	**Justification**
Timestep (Δt)	1 s	
Noise PSD (PSD_n)	60 dB re 1µPa^2/Hz	This is equivalent to a moderate sea state (Xerandy et al. 2015)
Source PSD (PSD_s)	120 dB re 1µPa^2/Hz at 1m	This is equivalent to the noise generated by a typical autonomous underwater vehicle (AUV) (Gebbie et al. 2012, Zimmerman et al. 2005)
Spatial step (Δx)	1000 m	Calculated based on the values of PSD_s and PSD_n.
Source centre frequency (f_c)	1 kHz	The central frequency of a typical autonomous underwater vehicle (AUV) (Gebbie et al. 2012)
Maximum velocity (U)	2 m/s	Typical autonomous surface vehicle (ASV) maximum speed range is 3 kn to 10 kn (Verfuss et al. 2019))
Signal propagation speed	1500 m/s	Speed of sound in water.
No. Robots	10	A typical swarm size used in marine robotics
Starting radius (d_0) and convergence radius (d_c)	50 m	Resulting in large enough area to accommodate 10 robots of typical ASV size
Forgetting function scaling parameter (a)	1	
Sensor separation (D)	0.75 m	Satisfies $D \leq \lambda_c/2$. This applies only to XB-PSO. Since Generalised Triangulation PSO only makes use of the average intensity of the signal, it only needs a single sensor at the middle of the robot
Non-Omnidirectional PSO parameter α	0.9	Slightly smaller than 1 to prevent collisions with other robots
Non-Omnidirectional PSO parameter β	0.9	A large value to ensure accurate control of the robots
Robot size	2×1m	A typical size for marine robotic platforms (Naval Technology 2021*a*,*b*)



(a) XB-PSO (b) Generalised Triangulation PSO

Figure 8-2: Percentile Distances of CoM to source at different SNR_0 for the tested algorithms. The solid lines represent the median route over 100 simulations, the 50% transparent areas represent the 50% percentile range and the 90% transparent areas the 90% percentile range.

out for 2000 timesteps (2000 seconds). This amount of simulation time is less than the simulation time used for the simulations of Chapter 7 (6000 timesteps) - this is because in these simulations the swarms make use of collision avoidance which improves the convergence performance of all PSO algorithms by keeping the robots far away from each other (i.e. it improves the Diversity Loss Problem found in all PSO algorithms), thereby needing less time to converge to the source. Each algorithm was run for 100 repetitions (the amount of repetitions needed to result in clear behavioural trends for both algorithms) and the results of Figure 8-2b show the median distance from the source, as well as the 50% and 90% percentile ranges at each timestep. The results show that Generalised Triangulation PSO is indeed capable of converging to the source. That said, in 2000 timesteps, it only manages to converge at 10 dB and 6 dB SNR_0, in contrast to XB-PSO which is capable of fast convergence at all SNR_0 values. This shows that Generalised Triangulation PSO suffers from the same range limitations as A-PSO; a result of using the intensity of the signal instead of higher-order statistics like B-PSO and XB-PSO. Therefore, it can be concluded that B-PSO and XB-PSO should still be used when it comes to wavefield sources and Generalised Triangulation PSO should only be used for the localisation of non-wavefield sources.

8.2 Source Entrapment/Escorting

With the definition of Generalised Triangulation PSO and the understanding of its limitations, it is now possible to study whether it is capable of allowing interaction with the source and localisation of a moving source.

In contrast to PSO, a robot in triangulation PSO does not remember the fitness/cost of its previous locations. Instead, the intersections of directional rays (and therefore the personal best locations) can be thought of as estimations of the location of the source based on information collected by the whole swarm and the best estimation is given by the global best location $\mathbf{y_g}$. Therefore, it may be possible for the swarm to interact with the source (e.g. avoid it), by interacting with the global best location. Furthermore, the algorithm makes use of a forgetting function that forces the robots to update their personal best locations as time passes. Therefore, if the swarm estimates correctly the location of the source and then the source moves, the swarm will estimate the new location of the source in the following timesteps, thereby keeping track of the source's movements.

Tasks that require such interaction with the source are the tasks of source entrapment/escorting, which are currently gaining a lot of attention in the research community (Antonelli et al. 2007, Kawakami & Namerikawa 2009, Zhang et al. 2018, 2020, 2021). In these tasks, the swarm is required to place itself around the source, each for different purposes (e.g. to prevent the source from escaping, to protect it from external intruders etc.). The main disadvantage of these methods is that they either assume that the swarm has prior knowledge of the location of the source, or they employ a very simplistic way of localising it. The inability of PSO (and other similar source localisation algorithms) to know the location of the source without passing on top of it has been the main reason why it has not been possible so far to merge collective source localisation and source entrapment/escorting algorithms.

In the next sections, it will be argued that this gap in the literature may be bridged using triangulation PSO. In their simplest form, source entrapment/escorting algorithms require the swarm to maintain a constant distance r from the source and to follow its movements. This can be simply achieved, by treating the source (or in the case of triangulation PSO, the global best location $\mathbf{y_g}$) as a large obstacle with radius r and using simple collision avoidance to maintain a constant distance from it. As long as this can be achieved, more complicated methodologies of source entrapment/escorting can also be used. To demonstrate these capabilities, further 2D simulations were performed in MATLAB. These will be discussed in the next sub-section.

8.2.1 Simulations

As in previous simulations, the source is assumed to emit a signal of constant power spectral density (PSD_s). The robots of the swarm make use of the average intensity of the received signal from a single sensor and Generalised Triangulation PSO is used to update the personal and global best locations of the swarm. The swarm's movement is once again controlled by the non-omnidirectional PSO controller, as described by Equation (8.7).

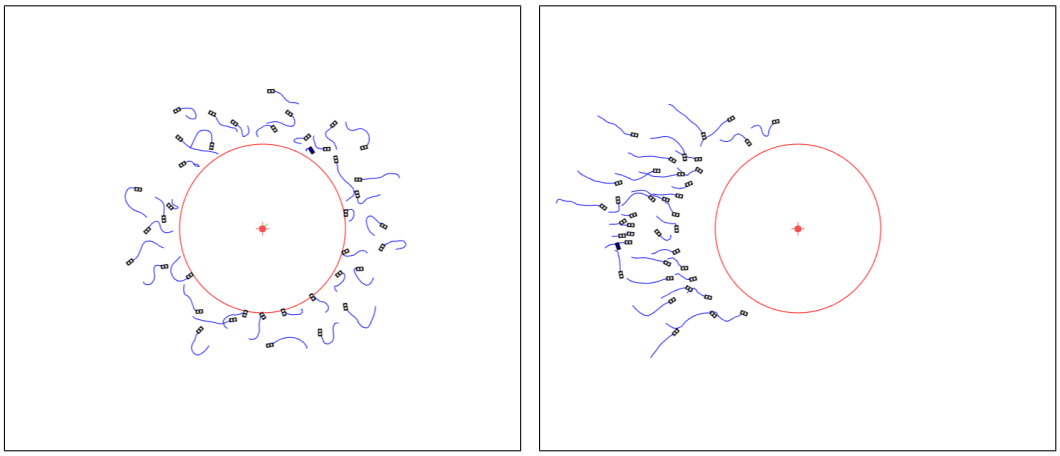
The following simulations will aim to demonstrate the capability of Generalised Triangulation PSO to interact with the source and study how this capability can be affected by several primary variables. The simulations are initially run for a stationary source and the swarm is tasked to maintain a distance r around it, by treating the global best location $\tilde{\mathbf{y}}_\mathbf{g}$ as an obstacle with radius r. Simulations are run for different values of r to study how the performance of the algorithm changes as the robots get closer to the source (i.e. in the near-field, where there exists larger curvature in the wavefront of the signal).

Furthermore, different simulations were performed for different numbers of robots in the swarm. Increasing the number of robots, increases the density of robots in a given area when the swarm is spread around the source, which can reduce the negative effects of wavefront curvature in the calculation of directional rays. Figure 8-3a, shows a single instance of the simulations where the source is stationary. The robots are spread around the source. The red ring represents the distance r around the source and whenever the robots get close to the ring, they "bounce off" of it.

Furthermore, simulations were performed for a moving source. In these simulations, the ability of the swarm to maintain a close distance from the source is tested. The velocity of the source is normalised with respect to the maximum velocity of the robots using

$$v_n = \frac{v_s}{v_{max}} \tag{8.8}$$

where v_n is the normalised velocity of the source, and v_s and v_{max} are the actual velocity of the source and the maximum velocity of the robots in units of m/s. Different simulations were performed for different values of v_n and r. Moreover, simulations were also performed for different values of signal intensities at source level to examine whether a change in the SNR of the received signals can affect the performance of the swarm. Figure 8-3b, shows a single instance of the simulations where the source is moving towards the right. The robots are shown to chase after the source, but they never get closer to it than distance r. Instead, they either slow down or they move towards the



(a) Stationary source (b) Source moving towards the right

Figure 8-3: Instances of the simulations carried out for the task of source entrapment/escorting. In the simulations, a swarm of robots (white rectangles) is tasked to maintain a minimum distance r, represented by the red ring, from the source (red target). In (a), the source is stationary while in (b), the source is moving towards the right with normalised velocity $v_n = 0.25$. In both simulations, the number of robots in the swarm is 40 and the power spectral density of the emitted signal at source level is 120 dB re 1µPa^2/Hz. The blue lines represent the path of each robot in the last 15 timesteps.

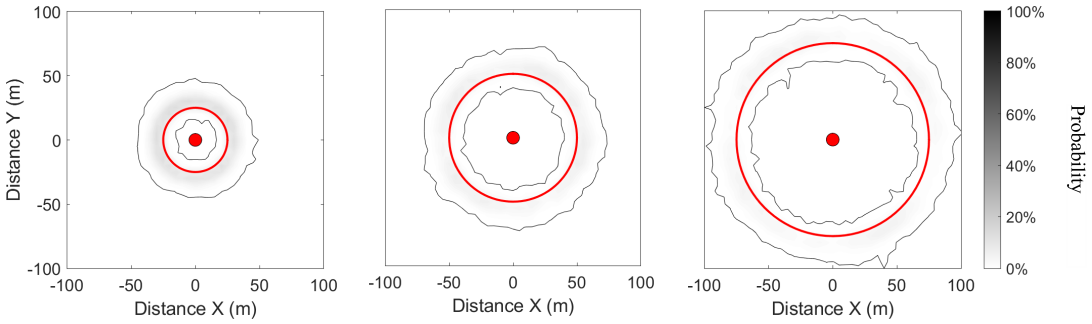
sides. The following sub-section presents the results of both the stationary source and moving source simulations.

8.2.2 Results

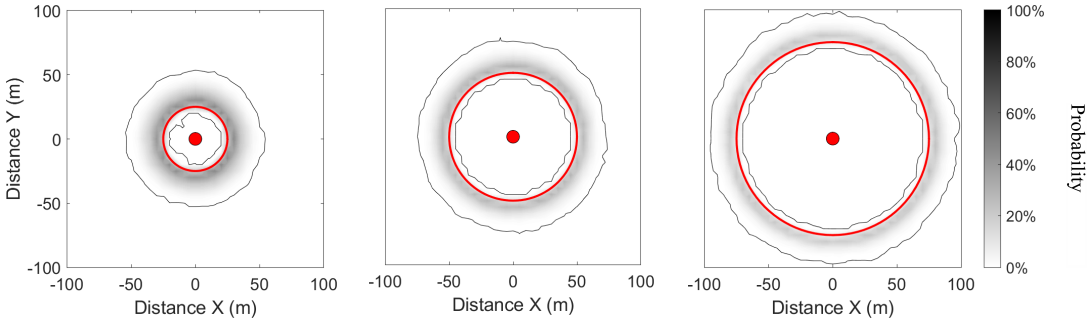
The following simulations were designed to replicate a marine robotics scenario. Therefore, the parameter values used will be identical to the ones shown in Table 8.1.

Stationary Source

Figure 8-4 shows the results obtained from simulations where the source is stationary. The number of robots N that form the swarm is 10 and 40 for different simulations (here $N = 40$ was selected because it clearly demonstrates significant differences in performance when compared to $N = 10$). The distance r that the swarm needs to maintain from the source varies (25 m, 50 m and 75 m), represented by the red ring. Each sub-figure shows the probability that a robot will occupy different regions of size 5×5 m at any given point, indicated by the shaded areas. The black outline around the shaded areas indicates the locations where there exists a 0.1% probability that a



(a) $r = 25\text{m},\ N = 10$ (b) $r = 50\text{m},\ N = 10$ (c) $r = 75\text{m},\ N = 10$



(d) $r = 25\text{m},\ N = 40$ (e) $r = 50\text{m},\ N = 40$ (f) $r = 75\text{m},\ N = 40$

Figure 8-4: Probabilistic distribution of the swarm relative to a stationary source (red dot). The source emits a signal with constant power spectral density PSD_s that the robots can detect. The swarm consists of N robots that are required to maintain a specific distance r (red ring) from the source. The shaded regions represent the probability that a robot will be found in a specific location at any given point. The black outline represents the locations where, there exists a 0.1% probability that a robot will be found there at any point.

robot will be found in that area at any given point. This outline shows how individual robots may behave differently than the rest of the swarm and it is used to identify how often a robot may get closer to the source than the desired distance r. Each sub-figure is the result of 20 simulations, which was the minimum number of simulations required to obtain statistical significance for the behaviour of the swarms. The resulting behavioural trends will be discussed below.

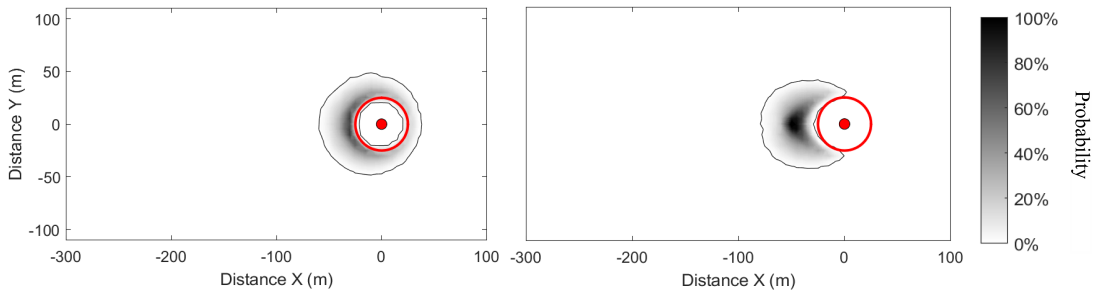
From the figures, it can be seen that when the number of robots in the swarm is small ($N = 10$), the swarm finds it hard to maintain the desired distance from the source. This can be seen by the 0.1% probability outline, which shows that robots can occasionally enter the restricted area indicated by the red ring. This is because the robots are spread around the source and therefore they are far away from each other. This results in incorrect calculation of the directional rays of the robots, which in turn can result in a noisy global best location $\tilde{\mathbf{y}}_{\mathbf{g}}$.

In contrast, a swarm of $N = 40$ robots is shown to be able to maintain well its distance from the source when $r = 50$ m and $r = 75$ m. Nevertheless, at $r = 25$ m, the swarm appears to not be able to maintain its distance properly. Since this happens only when the swarm is allowed to get close to the source, it is understood to be a result of Generalised Triangulation PSO being affected by the curvature of the wavefront of the signal (which is more intense the closer to the source).

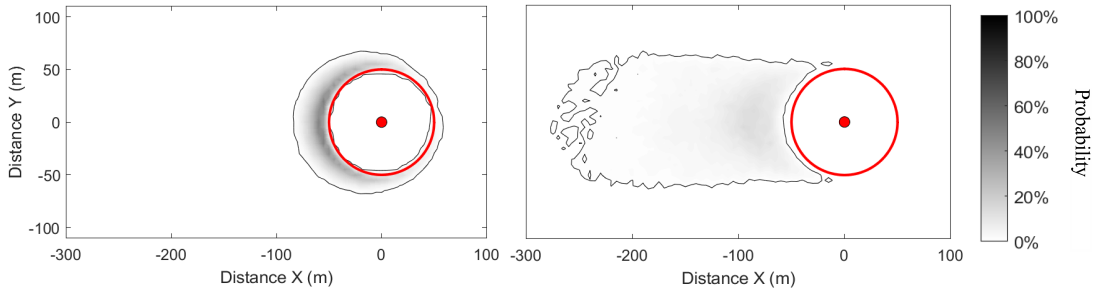
Therefore, the ability of the algorithm to maintain its distance from the source at all times is affected by the number of robots in the swarm and the desired distance that it needs to maintain. Note that other triangulation PSO algorithms that do not require the collaboration of robots for the calculation of directional rays (e.g. B-PSO and XB-PSO) should not be affected by changes to these parameters.

Moving Source

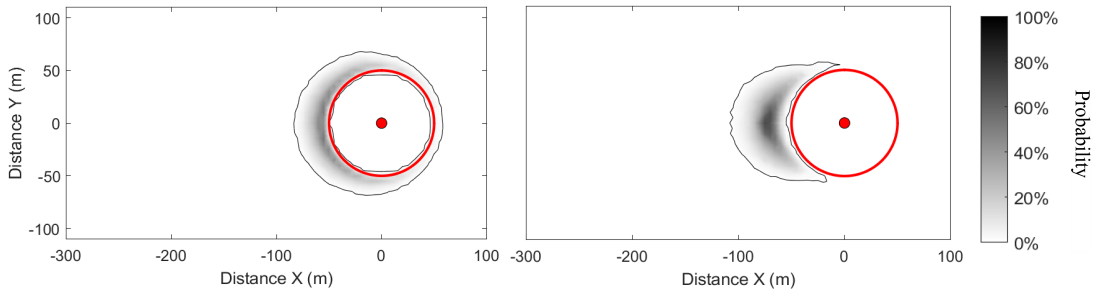
Figure 8-5 shows the results obtained from simulations where the source is moving. The number of robots N that form the swarm is 40 for all simulations and the distance r that the swarm needs to maintain from the source is 25 m and 50 m for different simulations. As in Figure 8-4, each sub-figure is the result of 20 simulations and it shows the probability that a robot will occupy a region of 5×5 m, indicated by the shaded areas. The black outline around the shaded areas indicates the locations where there exists a 1% probability that a robot will be found in that area at any given point. The reason why the outline in Figure 8-5 represents a different probability than in Figure 8-4 (1% instead of 0.1%), is that the simulations are initialised with the swarm spread



(a) $v_n = 0.1$, $r = 25$m, $\text{PSD}_s = 120$dB (b) $v_n = 0.25$, $r = 25$m, $\text{PSD}_s = 120$dB



(c) $v_n = 0.1$, $r = 50$m, $\text{PSD}_s = 120$dB (d) $v_n = 0.25$, $r = 50$m, $\text{PSD}_s = 120$dB



(e) $v_n = 0.1$, $r = 50$m, $\text{PSD}_s = 180$dB (f) $v_n = 0.25$, $r = 50$m, $\text{PSD}_s = 180$dB

Figure 8-5: Probabilistic distribution of the swarm relative to a source (red dot). The source is moving with normalised velocity v_n towards the right and it emits a signal with constant power spectral density PSD_s that the robots can detect. The robots are required to maintain a specific distance r (red ring) from the source. The shaded regions represent the probability that a robot will be found in a specific location at any given point. The black outline represents the locations where, there exists a 1% probability that a robot will be found there.

around the source. If a 0.1% probability was used, the swarms would appear to be able to move in front of the source which would be misleading. Furthermore, the purpose of these simulations is not to identify whether individual robots may occasionally enter the restricted area of the red ring but to show the general capability of the swarm to stay close to a moving source.

From the results it can be seen that when the normalised velocity of the source with respect to the maximum velocity of the robots is $v_n = 0.1$ (Figures 8-5a, 8-5c and 8-5e), the swarm is always capable of staying close and surrounding the source. This is the case, irrespective of the value of r or the SNR experienced by the robots. In contrast, when the normalised velocity of the source is set to 0.25, the swarm is shown to lag behind the source and it is not capable of encircling it any more.

The most interesting result is the case of Figure 8-5d, where $r = 50\,\text{m}$ and $\text{PSD}_s = 120\,\text{dB re } 1\mu\text{Pa}^2/\text{Hz}$. In this case, the swarm is shown to lag significantly behind the source. Individual simulations showed that as the swarm begins to lag behind the source, the lower SNR experienced by the robots causes the global best location $\tilde{\mathbf{y}}_\mathbf{g}$ to lag as well. This in turn pushes the swarm backwards, since the swarm avoids the global best location. Thus the swarm lags behind even more, reducing the SNR further and therefore causing a chain reaction that leads to the swarm losing contact with the source. Figures 8-5b and 8-5f show that the effect of this chain reaction can be eliminated by reducing the required distance that the swarm needs to maintain from the global best location or by increasing the signal intensity and therefore the SNR.

Generalised Triangulation PSO's ability to follow and surround a source, depends largely on the distance that the robots need to maintain from the source, the velocity of the source and the SNR of the received signal. Beyond these dependencies, it is highly probable that the dynamics of the non-omnidirectional PSO controller may introduce additional complexity to the system, making it difficult for the robots to travel towards the source at maximum speed. Future studies of this task could be performed with other types of controllers to study how the controller's dynamics affect the performance of the swarm. Furthermore, future studies could also include tasks like flocking and velocity consensus. Such tasks are meant to allow the swarm to travel collectively towards a specific direction, making it easier for it to build up speed and catch up with the source. It should finally be noted that the negative performance of Figure 8-5d could be minimised by using B-PSO or XB-PSO instead of Generalised Triangulation PSO, since they have much larger range and therefore are more robust to lower SNRs.

8.3 Conclusions

Triangulation PSO algorithms appear to offer significant advantages over PSO. Their ability to estimate the location of the source over distance allows them to interact with it. Furthermore, they are not limited to immutable environments and therefore they can be used to track moving sources. This chapter has introduced a variant of the triangulation PSO family that can be used for the localisation of non-wavefield sources. The algorithm's capabilities are demonstrated in simulations, where it is used for the task of source entrapment/escorting and it was shown that the swarms were capable of following and surrounding sources that are both stationary and moving. The algorithm's performance on these tasks seems to be affected by the distance required for the robots to maintain from the source, as well as the number of robots in the swarm, the velocity of the source and the SNR of the received signals.

References

Antonelli, G., Arrichiello, F. & Chiaverini, S. (2007), The entrapment/escorting mission for a multi-robot system: Theory and experiments, *in* '2007 IEEE/ASME international conference on advanced intelligent mechatronics', pp. 1–6.

Gebbie, J., Siderius, M. & Allen, J. S. r. (2012), 'Aspect-dependent radiated noise analysis of an underway autonomous underwater vehicle.', *The Journal of the Acoustical Society of America* **132**(5), EL351–7.

Kawakami, H. & Namerikawa, T. (2009), Cooperative target-capturing strategy for multi-vehicle systems with dynamic network topology, *in* '2009 American Control Conference', pp. 635–640.

Naval Technology (2021*a*), 'C-Enduro Autonomous Surface Vehicle'.
URL: *https://www.naval-technology.com/projects/c-enduro-autonomous-surface-vehicle/*

Naval Technology (2021*b*), 'REMUS-100 Automatic Underwater Vehicles'.
URL: *https://www.naval-technology.com/projects/remus-100-automatic-underwater-vehicle/*

Verfuss, U. K., Aniceto, A. S., Harris, D. V., Gillespie, D., Fielding, S., Jiménez, G., Johnston, P., Sinclair, R. R., Sivertsen, A., Solbø, S. A., Storvold, R., Biuw, M. & Wyatt, R. (2019), 'A review of unmanned vehicles for the detection and monitoring of marine fauna', *Marine Pollution Bulletin* **140**, 17–29.
URL: *https://www.sciencedirect.com/science/article/pii/S0025326X19300098*

White, K. G. (2001), 'Forgetting functions', *Animal Learning & Behavior* **29**(3), 193–207.
URL: *https://doi.org/10.3758/BF03192887*

Xerandy, X., Znati, T. & K, L. (2015), 'Cost-Effective, Cognitive Undersea Network for Timely and Reliable Near-Field Tsunami Warning', *International Journal of Advanced Computer Science and Applications* **6**.

Zhang, S., Liu, M., Lei, X., Huang, Y. & Zhang, F. (2018), 'Multi-target trapping with swarm robots based on pattern formation', *Robotics and Autonomous Systems* **106**, 1–13.
URL: *http://www.sciencedirect.com/science/article/pii/S0921889017306024*

Zhang, S., Liu, M., Lei, X., Yang, P., Huang, Y. & Clark, R. (2021), 'Synchronous inter-

cept strategies for a robotic defense-intrusion game with two defenders', *Autonomous Robots* **45**(1), 15–30.
URL: *https://doi.org/10.1007/s10514-020-09945-6*

Zhang, T., Liu, Z., Wu, S., Pu, Z. & Yi, J. (2020), Multi-Robot Cooperative Target Encirclement through Learning Distributed Transferable Policy, *in* '2020 International Joint Conference on Neural Networks (IJCNN)', pp. 1–8.

Zhao, F. & Guibas, L. J. (2004), 2 - Canonical Problem: Localization and Tracking, *in* F. Zhao & L. J. Guibas, eds, 'Wireless Sensor Networks', The Morgan Kaufmann Series in Networking, Morgan Kaufmann, San Francisco, pp. 23–62.
URL: *https://www.sciencedirect.com/science/article/pii/B978155860914350002X*

Zimmerman, R., D'Spain, G. L. & Chadwell, C. D. (2005), 'Decreasing the radiated acoustic and vibration noise of a mid-size AUV', *IEEE Journal of Oceanic Engineering* **30**(1), 179–187.

Chapter 9

Conclusions and Future Work

9.1 Conclusions

This thesis introduced the swarm control algorithms that would be required for the implementation of a swarm of marine robotic platforms with passive acoustic capabilities, for the localisation and monitoring of underwater acoustic sources. The presented algorithms achieve the following results:

- Accurate low-level motion control of the robotic platforms using Generalised Adapted PSO, by taking into consideration their movement limitations, such as maximum velocity, maximum acceleration, omnidirectional motion capabilities, turning radius etc. This allows the algorithm to be used directly for the control of the velocity of robots, instead of as a high-level trajectory planning algorithm like most current swarm intelligence algorithms.

- Proper synchronisation of the motion controller with the physical platform, by taking into consideration the loop delay of the controller (i.e. timestep size Δt). In this way, the controller consider delays introduced by high processing demanding tasks and inter-robot communications, thereby softening the limitations that exist on this areas and making the implementation of a real-world robotic swarm easier.

- Generalised Adapted PSO allows easy merging of a large number of off-the-shelf algorithms for different swarm robotic tasks and can be adapted to approximate different behaviours. This is demonstrated through the implementation of two PSO controllers (the omnidirectional and non-omnidirectional PSO controllers presented in Chapters 4 and 5, where the tasks of source localisation and obstacle

avoidance are combined.

- When it comes to signal processing, by incorporating wavefield correlation into PSO, its source localisation range and convergence speed towards the source were improved by a factor of 4 (compared to A-PSO).
- Additionally, the introduction of triangulation PSO allows the swarm to localise the source from far away. This capability removes the need for the swarm to be spread around the area where the source is expected to be found (a practice that is typically expected when using the original PSO algorithm), thereby making PSO more applicable to real-world scenarios, where large areas need to be explored.
- Finally, the ability to localise the source from far away, allows the swarm to interact with it, by avoiding, encircling and following it. This is a very important property of triangulation PSO algorithms that is crucial for the implementation of systems capable of source monitoring that is not found in other swarm intelligence algorithms that can be currently found in the literature.

Figure 9-1 shows which of the key resulting algorithms proposed in this thesis would be more appropriate for use in a robotic swarm, based on the robots' motion capabilities and the characteristics of the acoustic source that needs to be localised. All of the algorithms were demonstrated in MATLAB simulations, while the Omnidirectional PSO controller was also validated using more realistic Gazebo simulations. Since the rest of the algorithms were based on the Omnidirectional PSO controller, their validation using Gazebo was not required.

The resulting swarm behaviours that are achieved by the algorithms presented in this thesis consist the core behaviours required to be exhibited by a real-world robotic swarm for the tasks of source localisation and monitoring. This thesis therefore validates the possibility that such a real-world robotic system can exist from the control perspective, thereby opening the way for future attempts towards its physical implementation.

9.2 Future Work

Despite the validation of the core behaviours of the swarm using simulations, future work will aim to validate all proposed algorithms using real-world experiments, to further ensure their correct operation and applicability to physical swarms. Apart from the presented behaviours, several other derivative concepts will also need to be tested and validated in the future, to ensure that the proposed algorithms are capable of satisfying the control requirements of a robotic swarm for source localisation and monitoring.

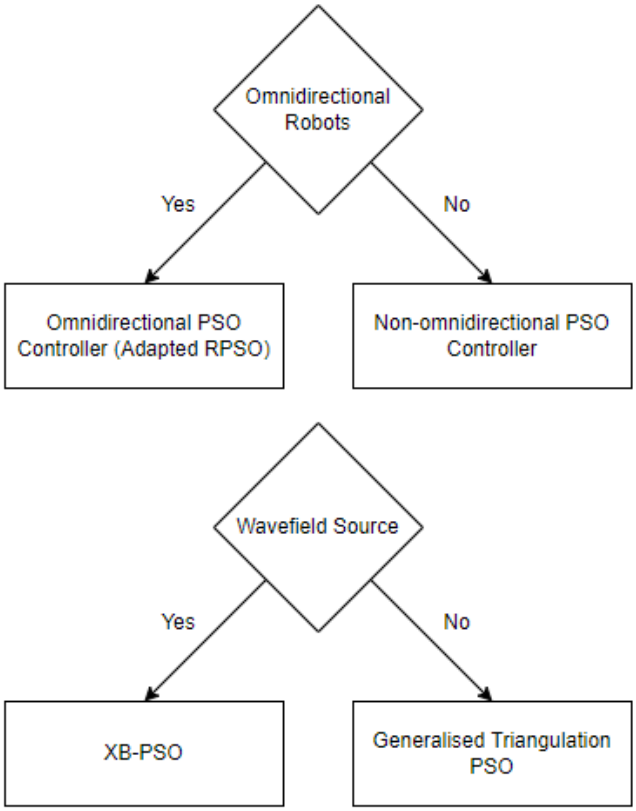


Figure 9-1: Flow chart that explains which of algorithms proposed in this thesis could be used for the control of a robotic swarm, based on the motion characteristic of the robots in the swarm and the source that needs to be localised.

Some of the most important concepts are outlined below in order of priority:

- The omnidirectional and non-omnidirectional PSO controllers offer a glimpse of how Generalised Adapted PSO could be used to merge different swarm robotic tasks and approximate different robot behaviours (i.e. omnidirectional and non-omnidirectional motion). That said, the possibilities offered by Generalised Adapted PSO are not limited to the tasks and behaviours used in this thesis. Tasks such as flocking and collaborative multi-target tracking can be of great benefit in source localisation and monitoring scenarios. Therefore, it is desirable to show how such tasks could be incorporated into Generalised Adapted PSO through the use of alternative Dynamic Velocity Control strategies. Additionally, showing how Generalised Adapted PSO could be used to control non-omnidirectional motion with minimum turning radius (as discussed in Chapter 5) can greatly extend the applicability of the algorithm to a larger variety of robotic platforms.

- With regards to the work presented in Chapter 8, the results are used to demonstrate the potential of using triangulation PSO algorithms to fuse collaborative source localisation and target entrapment techniques. The resulting behaviours validated the possibility of achieving such a fusion but the simulations did not actually use a target entrapment algorithm borrowed from the literature (instead the swarm interacted with the source by avoiding it like an obstacle). Therefore, future work on this area should study the swarm behaviours that emerge from the combination of triangulation PSO (either Generalised Triangulation PSO or another variant like XB-PSO) with other existing target entrapment techniques (e.g. the ones discussed in Chapter 2).

- The wavefield correlation enhanced PSO variants presented in Chapter 7 (i.e. X-PSO, B-PSO and XB-PSO) only make use of the non-normalised correlation coefficient ρ and time lag τ_{lag}, obtained from the cross-correlation of signals from *two* sensors. Apart from these basic signal processing techniques, wavefield correlation systems (e.g. multi-hydrophone linear arrays and multi-element antennas) employ a variety of more complex, derivative techniques (e.g. beamsteering techniques and more exotic alternatives of the cross-correlation function) that could be used to further enhance the performance of the proposed algorithms or introduce new modified variants.

- Lastly, the Dynamic Velocity Control strategies used for the omnidirectional and non-omnidirectional PSO controllers presented in Chapters 4 and 5, assume a constant timestep size Δt. Nevertheless, it is possible that a dynamic Δt could also be considered in the future. In cases where the robots are far apart from each other, a large Δt value could be used, implying less frequent communication between the robots, but also higher possible maximum velocities. As the robots come closer to each other (while converging towards the source), the value of Δt can be reduced, resulting in more frequent communications, lower maximum velocities and more accurate motion control of the swarm. Therefore, considering a dynamic timestep size can introduce additional functionality into the Generalised Adapted PSO controller.

Appendices

Appendix A

Adapted PSO Order-1 and Order-2 Stability

Like the original PSO, there is no interdependency between dimension in Adapted PSO. Therefore, if it can be shown that Adapted PSO is order-1 and order-2 stability for an single, it will imply order-1 and order-2 stability in all dimensions. Therefore, only a single arbitrary dimension j will be considered here.

Due to the use of the sgn function, the stability analysis for Adapted PSO is greatly simplified. It is possible to show that Adapted PSO is order-1 stable for certain values of c_1 and c_2. Equation (3.11) can be described using expected value terms as

$$E(u_j[k+1]) = \omega E(u_j[k]) + c_{1e} + c_{2e}, \tag{A.1}$$

where $E(u_j[k])$ is the expected value of the j^{th} component of the velocity at timestep $[k]$, c_{1e} is the expected value of the cognitive term with constant magnitude $|c_{1e}| = \frac{c_1}{2}$ and c_{2e} is the expected value of the social term with constant magnitude $|c_{2e}| = \frac{c_2}{2}$. When $c_1 > 0$ and $c_2 > 0$, the corresponding expected values c_{1e} and c_{2e} always point towards the personal best location y_j and global best location $y_{g,j}$ respectively.

When both the personal best location and the global best location are located towards the same general direction relative to the particle such that $y_j, y_{g,j} > x_j$ or $y_j, y_{g,j} < x_j$, the effects of c_{1e} and c_{2e} are added together resulting in the combine magnitude

$$|c_{1e} + c_{2e}| = \frac{c_1 + c_2}{2} = \frac{\hat{c}}{2}. \tag{A.2}$$

This will eventually cause the particle to move towards y_j and $y_{g,j}$. On the other hand when the particle is located in-between the personal best location and the global best location, such that $y_j > x_j > y_{g,j}$ or $y_j < x_j < y_{g,j}$, the combined magnitude of c_{1e} and c_{2e} becomes

$$|c_{1e} + c_{2e}| = \frac{|c_1 - c_2|}{2}. \tag{A.3}$$

Therefore, when $c_1 = c_2$, the combined magnitude of c_{1e} and c_{2e} is 0. In this case, the expected position of the particle $x_j[k]$ as $[k] \to \infty$ is given by the midpoint of y_j and $y_{g,j}$

$$\lim_{k\to\infty} E[x_j[k]] = \frac{y_j + y_{g,j}}{2}. \tag{A.4}$$

As c_1 becomes larger than c_2, the expected position moves towards y_j. Similarly, as c_2 becomes larger than c_1, the expected position moves towards $y_{g,j}$.

The above conclusions can be extended to the case where either $c_1 < 0$ or $c_2 < 0$. In this case, when the magnitude of the positive accelerating coefficient is larger than the magnitude of the negative accelerating coefficient, such that $\hat{c} > 0$, the effects of Equations (A.2) and (A.3) remain the same. Therefore order-1 stability can be ensured for any $\hat{c} > 0$.

As long as order-1 stability is guranteed (i.e. $\hat{c} > 0$), the particle oscillates in-between y_j and $y_{g,j}$ with varying amplitude of oscillations. For order-2 stability, it is necessary to show that the amplitude of oscillation does not exceed a constant value.

In Appendix D, it is shown that Adapted PSO has a maximum velocity limit U_j. Assume that the particle passes above one of the best locations (y_j or $y_{g,j}$) with velocity $u_j = U_j$, such that Equation (A.2) holds. Furthermore, assume that $\hat{c} = 0$, such that the decelerating effect of c_{1e} and c_{2e} is nullified. In this case, Equation (A.1) becomes

$$u_j[k+1] = \omega u_j[k]. \tag{A.5}$$

The total distance d_∞ that the particle overshoots as $[k] \to \infty$ is given by

$$d_\infty = \Delta t \times (U_j + \omega U_j + \omega^2 U_j + \omega^3 U_j + ...) \tag{A.6}$$

The geometric series shown in the parenthesis is known to converge at $\frac{U_j}{1-\omega}$ for $|\omega| < 1$ (Maor, 1987). Therefore, the maximum distance that the particle can overshoot is $d_\infty = \frac{\Delta t \times U_j}{1-\omega}$ for $|\omega| < 1$. As $\hat{c}$ increases, the expected deceleration of the particle during overshooting increases, reducing this maximum overshooting distance d_∞. Therefore, it can be ensured that Adapted PSO is order-2 stable for $|\omega| < 1$ and $\hat{c} > 0$.

A.1 References

Maor E. The Geometric Series. In: To Infinity and Beyond. Birkhäuser Boston. 1987. https://doi.org/10.1007/978-1-4612-5394-5_5

Appendix B

Lemma 1

Lemma 1. *For all* $\hat{\mathbf{z}}_j[k] \in \mathbb{R}^2$, $\hat{\mathbf{M}}\hat{\mathbf{z}}_j[k]$ *will always lie on the line given by:*

$$a_1(u, \omega, \Delta t) = \frac{\omega - 1}{\omega \Delta t} u \tag{B.1}$$

Proof. The matrix $\hat{\mathbf{M}}$ has eigenvectors

$$\mathbf{v}_+ = a \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \mathbf{v}_- = b \begin{bmatrix} 1 \\ \frac{\omega - 1}{\omega \Delta t} \end{bmatrix}, \quad a, b \in \mathbb{R}$$

and respective eigenvalues

$$\lambda_+ = 0, \quad \lambda_- = \omega$$

Matrix $\hat{\mathbf{M}}$ is diagonalisable, since it is of size 2×2 and has 2 distinct eigenvalues. Therefore, the column space of $\hat{\mathbf{M}}$ is fully described by the span of the eigenvectors that are associated with non-zero eigenvalues as shown below

$$\mathrm{C}(\hat{\mathbf{M}}) = \mathrm{span}(\{\mathbf{v}_-\}) \quad \text{for } 0 \leq \omega < 1$$

This implies that $\mathrm{C}(\hat{\mathbf{M}})$ is a line and its characteristic equation is given by

$$a = \frac{\omega - 1}{\omega \Delta t} u \tag{B.2}$$

and the vector $\hat{\mathbf{M}}\hat{\mathbf{z}}_j[k]$ will always lie on it. □

Appendix C

Lemma 2

Lemma 2. *For all* $\hat{\mathbf{z}}_j[k] \in \mathbb{R}^2$*, the vector* $\hat{\mathbf{z}}_j[k+1]$ *will always be located in-between the lines*

$$\begin{aligned} a_2(u, \omega, \Delta t, \hat{c}) &= a_1(u - \hat{c}, \omega, \Delta t) + \frac{\hat{c}}{\Delta t} \\ a_3(u, \omega, \Delta t, \hat{c}) &= a_1(u + \hat{c}, \omega, \Delta t) - \frac{\hat{c}}{\Delta t} \end{aligned} \tag{C.1}$$

Proof. The vector $\hat{\mathbf{b}}_j[k]$ is a vector of random magnitude and is always parallel to the line

$$a(u, \Delta t) = \frac{u}{\Delta t}$$

It has maximum length when

$$\hat{\mathbf{r}}_j = 1,\ \operatorname{sgn}(\hat{\mathbf{y}}_j[k] - \mathbf{x}_j[k]) = \pm 1 \implies \hat{\mathbf{b}}_j[k] = \begin{bmatrix} \pm\hat{c} \\ \pm\, \hat{c}/\Delta t \end{bmatrix} \tag{C.2}$$

Lemma 1 says that $\hat{\mathbf{M}}\hat{\mathbf{z}}_j[k]$ always lies on the line a_1 of Equation (3.24). When $\hat{\mathbf{b}}_j[k] = \begin{bmatrix} \hat{c} \\ \hat{c}/\Delta t \end{bmatrix}$ as shown in Equation (C.2), the vector $\hat{\mathbf{z}}_j[k+1]$ must lie on the line

$$a_2 = \frac{\omega - 1}{\omega \Delta t}(u - \hat{c}) + \frac{\hat{c}}{\Delta t}$$

Conversely, when $\hat{\mathbf{b}}_j[k] = \begin{bmatrix} -\hat{c} \\ -\hat{c}/\Delta t \end{bmatrix}$, the vector $\hat{\mathbf{z}}_j[k+1]$ must lie on the line

$$a_3 = \frac{\omega - 1}{\omega \Delta t}(u + \hat{c}) - \frac{\hat{c}}{\Delta t}$$

Therefore, in all other cases, the vector $\hat{\mathbf{z}}_j[k+1]$ must always be located between the lines a_2 and a_3. $\square$

Appendix D

Theorem 1

Theorem 1. *For all ω, $0 \leq \omega < 1$ and $\hat{c} > 0$, there will always exist a maximum velocity $U_j \geq 0$ such that $|u_j[k]| \leq U_j$*

Proof. To find the value of U_j, let a robot accelerate in a single direction so that,

$$u_j[k+1] = \omega u_j[k] \pm \hat{c} \tag{D.1}$$

In this case, Equation (D.1) represents a non-homogeneous first-order linear recurrence relation. Assuming that $0 \leq \omega < 1$ and $\hat{c} > 0$, the maximum velocity U_j is given by

$$U_j = \lim_{[k]\to\infty} |u_j[k]| = \frac{\hat{c}}{1-\omega} \tag{D.2}$$

Therefore, if a robot is allowed to accelerate as much as possible towards a specific direction, its velocity will asymptotically approach U_j resulting in $|u_j[k]| \leq U_j$ for any value of $[k]$. □

Appendix E

Theorem 2

Theorem 2. *For all ω, $0 \leq \omega < 1$ and $\hat{c} > 0$, there will always exist a maximum acceleration $A_j^+ \geq 0$ and a maximum deceleration $A_j^- \leq 0$.*

Proof. The maximum acceleration can be found by setting $u_j[k] = 0$ in Equation (3.22) and assuming that the sgn function is positive, resulting in

$$A_j^+ = \frac{\hat{c}}{\Delta t} \tag{E.1}$$

Conversely, the maximum deceleration can be found by setting $u_j[k] = U_j$ in Equation (3.22) and assuming that the sgn function is negative, resulting in

$$A_j^- = \frac{(\omega - 1)U_j - \hat{c}}{\Delta t} \tag{E.2}$$

Substituting Equation (D.2) in Equation (E.2) results in the relationship between A_j^+ and A_j^-

$$A_j^- = \frac{-2\hat{c}\sqrt{d}}{\Delta t} = -2A_j^+ \tag{E.3}$$

Equations Equation (3.27) and Equation (E.3) are well-defined expressions of A_j^+ and A_j^- in terms of ω, $\hat{c}$ and Δt, under the only conditions that $0 \leq \omega < 1$ and $\hat{c} > 0$. Therefore, under these conditions, $A_j^+ \geq 0$ and $A_j^- \leq 0$. □