MAPPING THE OCEANS USING FLOCK DISTORTION

Ehsan Honary[1], David McFarland[2] and Chris Melhuish[3]

Intelligent Autonomous Systems Lab
University of West of England, Bristol
[1] ehsan.honary@uwe.ac.uk
[2] david_mcfarland_uk@yahoo.com
[3] chris.melhuish@uwe.ac.uk

Abstract

Currently no equivalent system of GPS exists for underwater applications. There is therefore a need to track objects underwater by other means. Here we report on a novel algorithm which employs a flock of underwater robots to generate depth profiles without the aid of marker communication systems. In this approach only inter-robot distances are required. Here, the sampling flock can map over long distances and is not constrained to the marker communication range. This paper demonstrates a method for calculating the positions of a flock of underwater robots using only the recorded relative positional information of the robot flock. The calculated absolute path of the underwater robots can then be used to map the sampled sensory information such as CTD in 3D. The flock is dropped into the ocean and is not tracked by any external device until it resurfaces later on. The data is collected from the 'surviving' robots and a novel algorithm is applied to this data to calculate the absolute 3D trajectory taken by the robots over the duration of the sampling. In this paper, the *flock estimation algorithm* and the associated issues of generating a velocity profile are presented and discussed.

Introduction

There has always been a need in oceanography to monitor the ocean currents and sample the underwater environment. Over the years various methods have been developed. These fall into two main categories:

- *Lagrangian measurements.* Currents are tracked by objects within the water mass. These objects are referred to as *Lagrangian subsurface floats* or *Lagrangian drifters* and are used to measure the movement and velocity of the currents.
- *Eulerian measurements.* Currents are monitored in a particular location of the ocean. Sonar, remote sensing and fixed Acoustic Doppler Current Profiler (ADCP) are in this category. For example, this technique is used when there is a need to know the current velocity at a particular place, such as under a bridge or opening to a bay.

The method presented in this paper fits into the category of Lagrangian measurements. There are currently many varieties of buoys used for this type of measurement, which are successfully commercialized (Davis et al. 1991).

The method explained in this paper has certain benefits in comparison with conventional methods of sampling environmental variables. Since the flock of robots do not need to be in range with an external marker for positioning, they can be used in situations where either it is costly to use external markers or where it is physically difficult to do so. For example a flock of robots in conjunction with our flock distortion technique can be used under ice caps to measure the currents and the environmental variables. The flock can also generate more accurate current profiles as the trajectories of the robots are estimated underwater. Since the majority of the calculations in the *flock estimation algorithm* are processed in a base computer, the extra overhead cost is small. The algorithm presented in this paper requires the use of inter-robot distances. Such an algorithm could therefore be employed using slightly modified existing underwater vehicles (e.g. Argo floats).

Perhaps one of the best examples for Lagrangian measurements is the Argo project (Roemmich et al. 1999), which aims to observe the oceans globally. It uses a global array of autonomous profiling floats. The resolution of this system is around 200-300 km, which is useful for climate and oceanography scientists. The system consists of floats that are similar to PALACE probes or *Profiling Autonomous Lagrangian Circulation Explorer*, which was developed by Woods Hole Oceanography Institute (Davis 1991).

These probes are usually dropped into the ocean from a surface ship. After deployment, they descend to a predefined depth (around 2000m). The robots drift in the currents. They sample the ocean for temperature and salinity during the drift. After a certain time, they ascend and transmit their data through a satellite to some land base. The collected data is then calibrated and evaluated before it is supplied to the scientific community. This cycle is repeated for each float. The floats operate around 3-4 years (Davis 1991). Each cycle is approximately 14 days. The floats remain almost a day on the surface for transmission of data. The movement of the floats is graphed using lines for each cycle of the floats. The interconnected lines show the movements of the floats caused by the underwater currents.

Argo is a good system for understanding and observing the oceans in a truly global manner. However, the resolution of the sampling is not high with this system. This paper addresses the problem of high-resolution mapping of environmental variables.

The observed movement of the Argo floats is based on the assumption that the floats move in straight lines between each resurfacing point. However, if at any point more resolution is required then linear interpolation cannot be used successfully. This also applies if only a certain area of the ocean needs to be observed accurately. In these cases there is a need to know the location of the probes underwater at all times. Therefore, by using their trajectory over time, it is possible to map the current velocity profiles along with other environmental variables sampled during the run.

Flock Distortion

Flock distortion, the mechanism proposed in this paper, can be used to detect the position of the robots underwater. Basically, the relative deformation of the flock, as it sinks, is used to calculate the absolute trajectories of the individual robots underwater. These robots are capable of only moving up and down by themselves and will only drift in the horizontal direction. These robots have the same functionality as the floats used in the Argo project, though they are used in conjunction with flock distortion technique.

Flock distortion is used as follows: A flock of robots are dropped into the ocean from a ship or an aeroplane (Figure 1). The robots hit the water at a particular location. They are constructed in such a way to float for a few seconds before they start to descend. In this time they can record their initial locations using GPS. This information is saved in each robot. The robots then descend under the force of gravity. All the robots are synchronized in time, and at each time step they can record the input from their sensors. They will record the depth, the salinity of water, the water temperature, and so on. In addition, each robot will record the distance between itself and all other robots that are in range with it. This can be done by using sonar with different codes so that the robots can have separate identities. This technique does not require for each robot to record the bearing between itself and other robots. This makes the robots a lot simpler and cheaper to design and manufacture.

The cycle of operation is as follows: As each robot descends it records the required environmental information, and the distances to the other robots until it reaches a certain depth. This depth is set by trimming the robot in such a way as to have the same density as the surrounding seawater at this depth. The robots stay at their trimmed depth for a while (which can be different from each other) and drift with currents. After a certain amount of time (set by the user), the robots start to ascend. On the way up the robots still continue to record the information. It is likely that the robots will be ascending through different areas of the ocean, so the data recorded during the ascent is just as useful as that taken during the descent. When they reach the surface of the water, the robots locate themselves using GPS and then transmit their position along with the rest of the collected data to a land-based computer via a satellite. The robots can be collected from the surface of the water, or they can be abandoned if collection is considered too expensive. During the run there is no need for marker communication, surface pods or ships. Once the data is collected (referred to as *distance table*) the *flock estimation algorithm* is applied and the results are obtained. This is subject to certain requirements, discussed later in this paper.

To control the flocking of the group it is important to control the buoyancy in order to keep the relative vertical positions of robots constant. The robots can only control their motion in the vertical dimension. So the flocking algorithm can be applied only to keep the group of robots together in respect with one dimension. Using the depth and distance measurements and applying a flocking algorithm, it is possible to achieve this flocking behaviour. This is discussed later in this paper.

For the purpose if this research, a new robot mechanism was developed, which is called Divebot. There were two main reasons behind the development of this robot. First it uses a novel mechanism in controlling its depth, which is explained shortly. Second, it was realised that during the development of the algorithm, it is important to be able to simulate the robots'

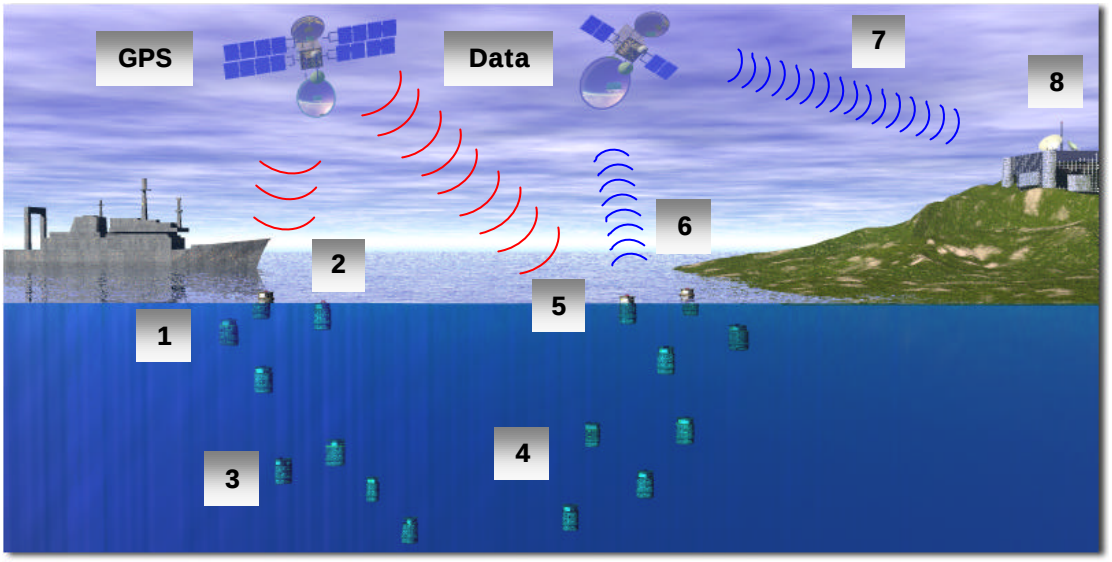

Figure 1: The components of the system. 1: The robots are deployed into the ocean. 2: The robots record their GPS location. 3: The currents underwater affect the robots. 4: At a predefined time set by the user, the robots start ascending. 5: Robots record their final GPS location. 6: Robots send the collected data to a satellite. 7: The satellite relays the data to a land base or back to a ship. 8: In the base, data is collected, stored and is processed.

performance underwater. This is useful for creating realistic simulations of the absolute trajectories taken by the robots, which can then be used to exploit the weaknesses of the algorithm. If for example, the robots are not simulated properly and are assumed to descend with a constant terminal velocity, then the relative vertical position of the robots stays constant. As a result the algorithm produces perfect trajectories with minimal error. However, it is unlikely that the robots will have the same terminal velocity in reality and hence the dissimilarities of the robots would lead to eventual error in the calculations. Analysing this error is quite important, so as a result a proper simulation (one that could at least create such dissimilarities) was required for the development of the algorithm. The *flock distortion technique* proposed here could, of course, be employed by existing Lagrangian devices such as Argo floats.

Divebot

The Divebot (developed in the IAS laboratory) has a buoyancy mechanism based on sperm whale (Clark 1979). The conventional buoyancy mechanism is usually based on pumps and pistons. In contrast Divebot heats the oil placed in its *oil chamber* to control the buoyancy. The oil chamber is contained within an insulated cylinder. Divebot mechanism is explained in detail in (McFarland *et al*., 2002). The picture of the prototype is shown in Figure 2. If the oil is heated, it expands and pushes some seawater out which reduces the overall mass of the robot. This way the Divebot can come up to surface. When the oil is cooled down (by turning the heater off), seawater is pulled in and the Divebot sinks. By switching the heater on and off it is possible to control the sinking rate of the Divebot. Since there is a time lag for heating oil, the control is not spontaneous. However, this is not really a problem, as the main purpose of buoyancy control is to stay in the flock and to be able to come up to the surface in the end of the run.

Since the Divebot is oil-filled, it has the advantage of withstanding far more pressure than conventional pressure vessel systems and so it could be a robust alternative for deep-ocean applications. The instrumentation on Divebot can be very similar to floats used in the Argo project. The current prototype has sensors for salinity, depth and sea temperature. It can also log the data collected on board, which can be transferred to a computer when the robot resurfaces.

To understand the dynamics of the Divebot an enhanced computer simulation was developed. Because of space limitation, a fuller description of the mathematics employed by the simulation will appear in a later paper.

To simulate Divebot behaviour, an abstract three-dimensional ocean is first simulated. A set of equations are used to calculate the density and the pressure at any depth and location underwater based on the temperature and salinity at that point and also the density of the water above. However, the ocean simulation is practically a one-dimensional simulation. This means that if any region has a higher density, the water flow from there to a region

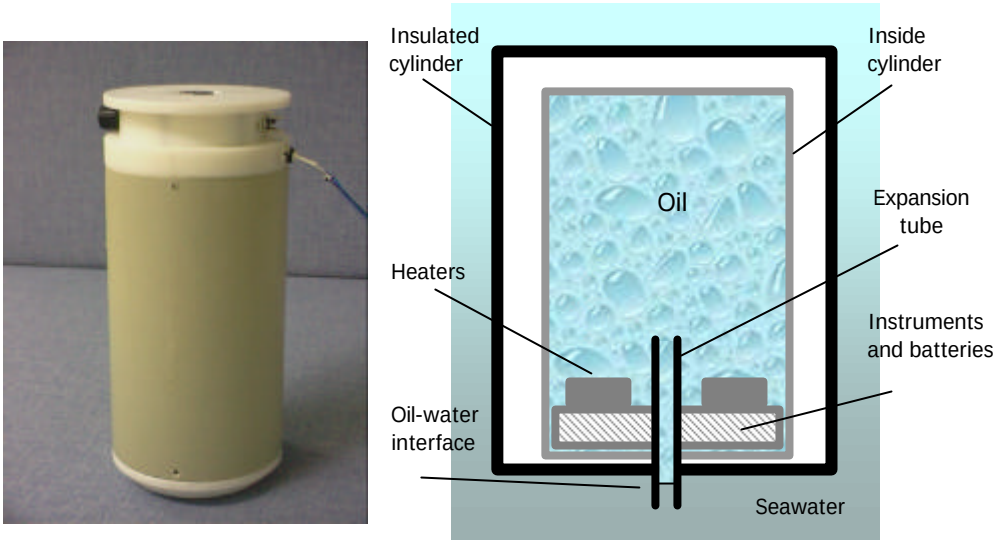

Figure 2: Divebot prototype (Left) (Length: 38 cm, Diameter: 16.8 cm, Weight: 8.4 Kg). Internal operation (Right).

with a lower density is not simulated. This is not necessarily a problem as the most important aspect of Divebot simulation is the vertical movement and not horizontal. The currents are simulated separately.

Once the characteristics of the ocean are simulated, the Divebot is placed in this environment and the forces applied to Divebot over time are calculated. There are three main forces: Gravity force applied to Divebot, which is basically weight (downwards), Buoyancy force (upwards) and the drag force (against movement). Once the overall force is calculated, it is the possible to calculate the acceleration, speed and position of Divebot over time.

These three forces are all variable. The buoyancy force depends on the density of the seawater around the Divebot which itself depends on sea temperature and salinity. The drag force depends on the hydrodynamic shape of Divebot, the velocity of Divebot in water and the seawater density at that point. The gravity force applied to the robot is the most complicated to calculate. It depends on the mass of the insulator and instruments and the oil, the *sinking mass* and also on the amount of variable seawater inside the chamber. This variable seawater depends on how much oil has expanded and pushed out water. So once the volume of oil is calculated, by using the density of the seawater inside it is possible to calculate the mass of seawater and eventually the total weight of Divebot. The sinking mass is the mass added to the trimmed Divebot to make it sink. By varying this value it is possible to control the sinking rate of Divebot and the eventual sampling depth in which the density of Divebot is the same as its surrounding seawater.

The volume of oil depends on the temperature of oil at that time and the oil expansion characteristics. The temperature of oil is calculated using standard thermodynamics formulae which takes into account heat capacity of oil, insulation characteristics and the amount of energy transferred into the oil from the heaters over time. The heaters are switched on and off by the control module of Divebot that creates the flocking behaviour. The batteries and current consumption of heaters and instruments are also simulated. This way it is possible to see when Divebot runs out of energy and what trajectory it will take once this happens. Figure 3 shows the simulation results for the prototype version of Divebot. Unfortunately the prototype Divebot cannot go very deep with its current implementation. Therefore the simulation example is setup in such a way that the robot would stabilise around 100m. To achieve this, 10g of sinking mass is placed on top of it. The robot stabilises at depth of 112m, when its density is the same as the seawater around it. The Divebot is programmed to switch the heaters on after one hour to bring it up to the surface.

It is important to notice that the simulation is capable of employing many different varieties of buoys and hence is possible to find the best characteristic of the ultimate Divebot used in real-life applications. The performance of the current prototype can be improved greatly by changing the shape, the type of oil used, the insulation material and higher capacity batteries. These issues will be discussed in a separate paper.

Flock Simulation

The simulation of the flock is accomplished by using the Divebot and the ocean currents sub-

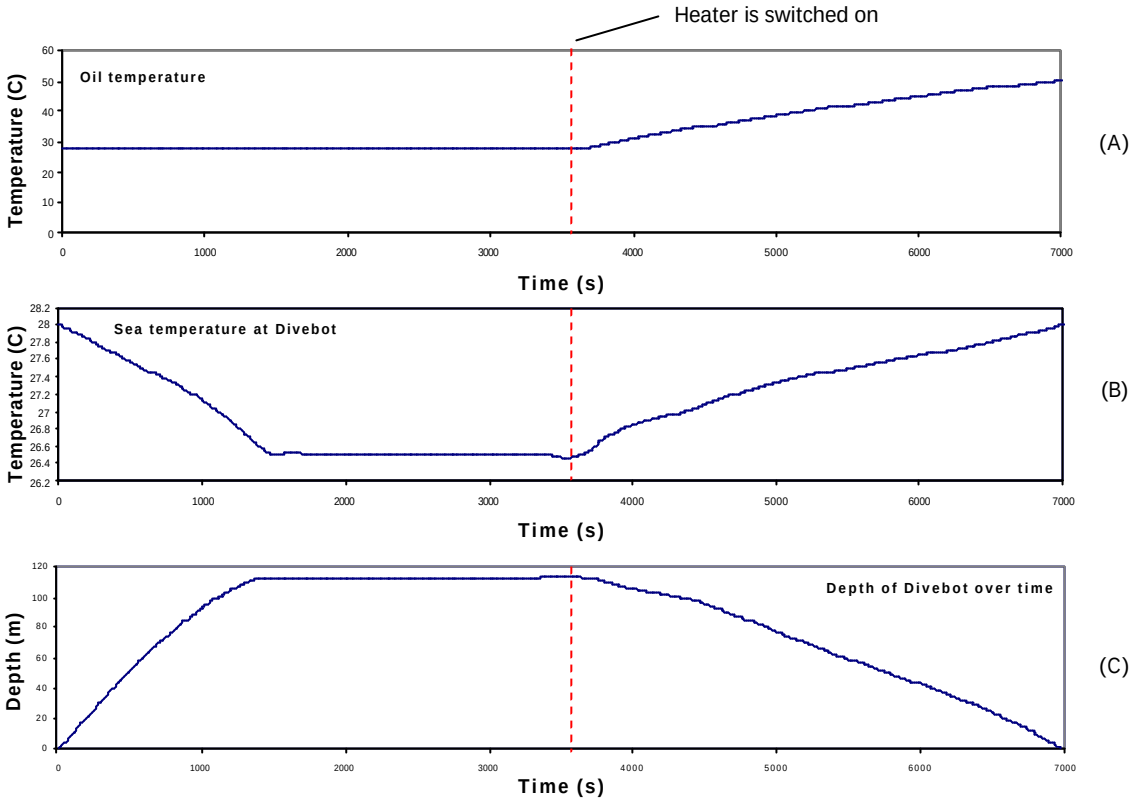

Figure 3: Some of the results of Divebot simulation. Here the oil temperature of Divebot (A), the sea temperature at Divebot's depth and location during the sampling (B) and the depth of Divebot (C) over time is shown. The Divebot is programmed to switch the heater on after one hour (indicated by dashed line). Notice that there is a lag between the time that the heater is turned on and the time that Divebot starts to rise.

programs in the simulator. The currents are simulated separately from the ocean parameters as explained before. In this study, it was adequate for the currents to be represented by vectors at different locations in the 3D space. These vectors can then influence the motion of the flock to create relatively realistic deformations. As a result the currents and the ocean were simulated independently. The algorithms, the simulation and all the software for this research were developed using C#, XML and the .Net Framework.

The flock simulation uses certain parameters chosen by the user for the initial deployment of the robots. The area under consideration, the distances between the robots, the shape of the flock and the timing between dropping each robot into seawater is chosen and the flock is then simulated to obtain their absolute path underwater. The robots also record their depth and distances between each other. Range is also simulated. This means that if two robots are further away than a certain amount, they will not be able to record the distance between each other. The simulation is also able to apply any uncertainty model to the recorded distances. For example Gaussian noise can be added to the recorded distances to test the *flock estimation algorithm* to see if it can cope with realistic uncertainties.

In reality the distances are recorded by sonar. There has been considerable research in this area and as a result there are many commercial products available for this purpose, though all with certain advantages and disadvantages. For example, in 1974 a series of experiment were accomplished on using Minimode tracking system to simultaneously track multiple floating buoys from a ship using sonar (Swallow 1974). The distances recorded by robots in the system presented in this paper is based on the same principle, though with one great advantage. While in the Minimode system, the robots are all tracked from the ship, in flock distortion technique, the robots track each other as they go along without the necessity of staying in range with a ship or a marker. This increases the accuracy of the distance measurements as the robots are usually closer to each other than they ever would be to an external marker. In this paper the concern is how to extract absolute information from the relative collected distances. The choice and issues associated with distance measurement device is subject of future research.

For the subject of this research different ocean currents are divided into four main classes:

- *Class 1: Constant horizontal currents***.** These currents are horizontal. At a certain depth in the ocean there is a particular value for the velocity of the current and this is constant for the whole ocean. The currents do not change over time, and there are no vertical currents.
- *Class 2: Varied horizontal currents***.** These currents are the same as Class 1 except that they can be different at different positions in the ocean. This means that robots dropped in different positions in the sea might experience different currents.
- *Class 3: Varied horizontal currents with varied vertical element***.** These are the same as Class 2 and in addition they have vertical elements added to the currents. This means that at any (x,y,z) position in the ocean there can be a (V_x,V_y,V_z) vector that represents the current.
- *Class 4: Varied horizontal currents with varied vertical element that can change over time.* As the title suggests, the velocity vector can have different values at different times. This is the same as real world currents.

In this paper mainly *class 1* and *class 2* currents are considered. The higher classes require more sophisticated algorithm to handle them and are the subject of future research.

The results described in this paper essentially reflect the 3D nature of the problem domain. This is difficult to represent visually and so sets of 2D views of resulting trajectories are shown in order to illustrate 3D trajectories. The illustration shown in Figure 4 is the basis behind all the 3D graphs shown in this paper. Figure 5 shows a flock of Divebots dropped into the ocean, where the Divebots are unique. This means that each of them has slightly different characteristics from another. For example some of them have a heavier mass than others. This makes them sink faster and come up slower simulating the inaccuracies of mass manufactured robots. In this figure a *class 1* current is applied to the flock. Figure 6 shows the depth of Divebots over time.

It is easy to see that the shape of the flock changes as the flock experiences the current. Also since each Divebot is unique, the resurfacing location is different for all of them as some of them are slower to ascend and hence stay in the current for a longer time and drift further than others.

Flock Estimation Algorithm

Once the flock is simulated, the recorded distances and the depth are retrieved from the robots to form a *distance table*. In addition the original and final resurfacing locations of the robots are recorded too. The distance table is the only information fed into the flock estimation algorithm. The structure of the *flock estimation algorithm* is as follows:

1. *Distance table processing*. This will traverse the distance table to fill in any field left without data. For example if the recorded distance by robot 1 between itself and robot 2 is different from the distance recorded by robot 2, these values are averaged to enhance the distance table. Also calibration charts and other sensory enhancements can be performed at this level.
2. *Seed robots search algorithm.* This searches for the choice of three *seed robots* using the *distance table*. The *seed robots* are basically the main 'internal' reference robots used for the estimation algorithm. The choice of these can be changed during the sampling, but they should be known throughout the rest of the calculations (parts 3,4 and 5).
3. *Relative estimation algorithm.* Using *seed robots* and the *distance table*, this algorithm calculates the relative path of all robots. At any given point in time only two of the three *seed robots* are used as reference robots.
4. *2D/3D velocity profile estimation algorithm.* Using the results of the previous algorithm, the velocity profile and the absolute path of all three *seed robots* are calculated.
5. *Absolute trajectory estimation algorithm.* Using the results of *seed robots*, the absolute path and velocity profile of all other robots are

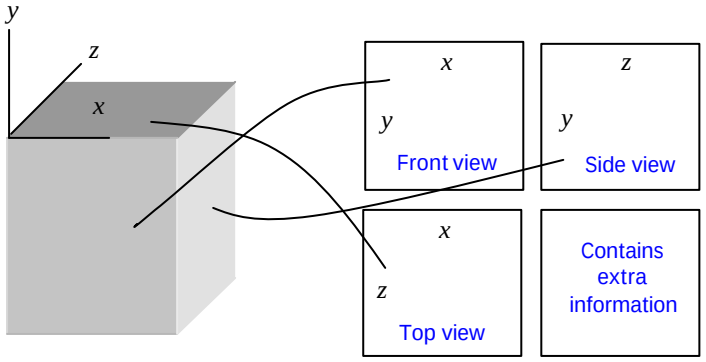

Figure 4: The 3D environment is shown using the three standard 2D planes. Notice that the vertical axis is *y* as opposed to *z*. This is chosen due to the standards used in the 3D graphics industry and hence easier implementation of the algorithm. In some figures the bottom-right box might contain extra information about the simulation.

calculated. This also applies any enhancements required and checks for the validity of the results.

The *Seed robots search algorithm* is perhaps best explained after explaining the rest of the algorithm. What follows is the core of *flock estimation algorithm.* Notice that in the following descriptions, triangles and points are all mapped to the surface plane as the depth of the robots are known. This makes the calculations easier to implement as they are executed in a 2D plane.

Relative estimation algorithm

The purpose of this algorithm is to create a relative trajectory of the path of robots underwater. Since the original locations of the robots are known (from recorded GPS coordinates), it is possible to calculate the position of each robot in relation `to two reference robots selected in the *Seed robot search algorithm.* This is accomplished by calculating a series of triangulations. Therefore the relative results for the flock has the same shape as the simulated or real flock, but not in the same location in relation to the world. The actual maths behind the calculations of triangles is explained in (McFarland and Honary 2002). Since the depth is known, the robots are mapped onto the surface of the ocean and all the triangles are calculated in this plane. The structure of this algorithm is as follows:

- Calculate the first position of each robot using the recorded original positions from the distance table
- For each measurement:

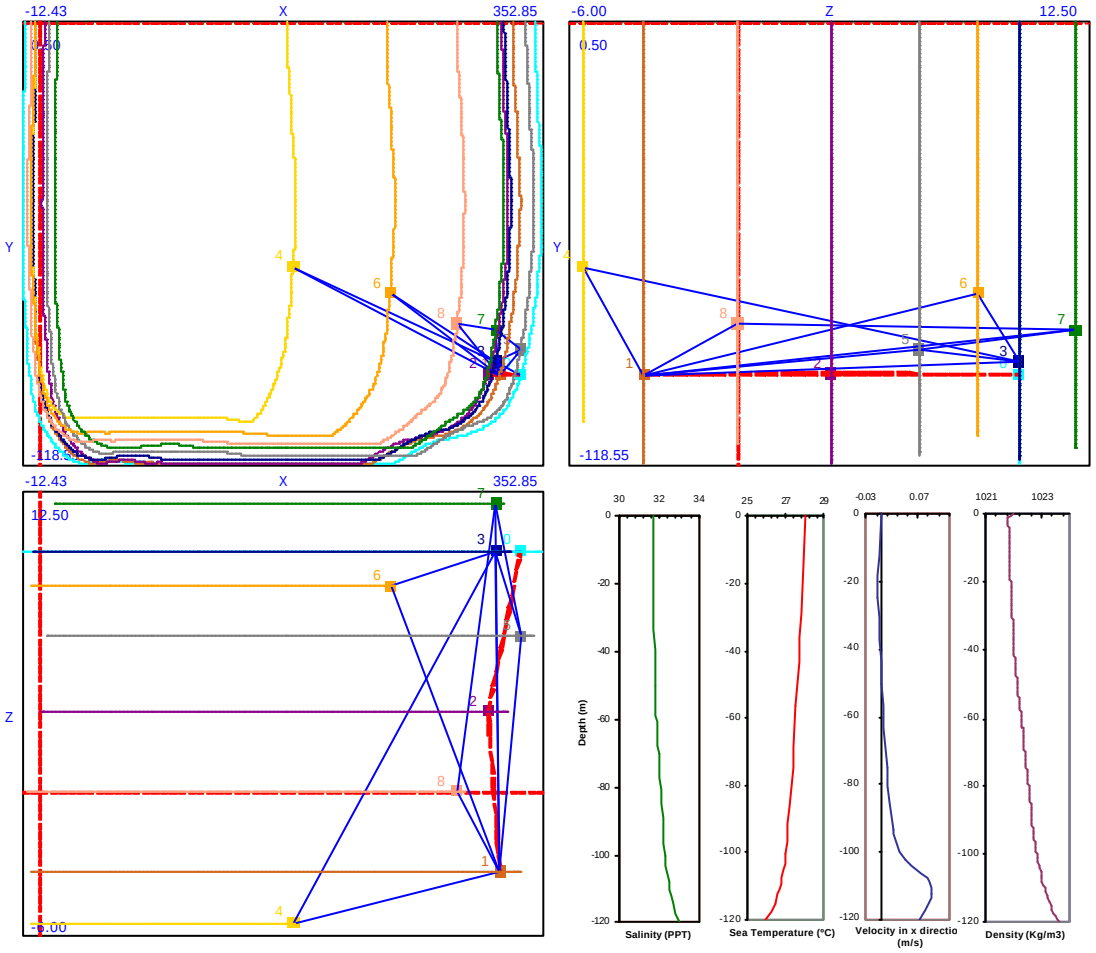

Figure 5: Simulation of a flock of Divebots shown in three separate planes for 3D (as described in Figure 4). Notice that the scales are not uniform in both directions in each plane. The vertical and horizontal dashed lines are the axis. The numbers on the sides of the boxes are the coordinates of each 2D plane in meters. The solid rectangles are the robots and each have a unique number next to them identifying the robot. The jagged lines are the trajectories taken by the robots underwater over time. Basically the robots are dropped in the ocean and are programmed to come up to surface after one hour of operation. Surface of ocean is at depth 0. Depth (y axis) is negative when underwater. The blue links between robots (solid rectangles) represent which triangles were used for position calculations. The Red links represent the first triangle solved between the seed robots, which are explained later. The salinity, sea temperature and velocity provided to the simulation are shown on bottom right. Density is the result of the simulation, which along with other parameters is used to simulate the flock.

- Use the seed robots array (the product of *Seed robot search algorithm*) to choose the *reference robot* and the *direction robot*
- Calculate the direction of the *direction robot* in relation with *reference robot* which is always in the same direction as its original position
- Calculate the position of *direction robot* using the recorded distance between the *reference* and *direction robot*
- If the *reference robot*, the *direction robot* and another robot are underwater, then carry on with the calculations. Otherwise use the original locations of all robots as their position.
- Solve the first triangle. Two points of this triangle are *reference* and *direction robot*. In order to find the third point of this triangle, search the distance table to find a robot that is in range with both the *reference robot* and the *direction robot*.
  - If this robot cannot be found, then use previous points for this measurement as there is no solution
- Add these three calculated robots to the list of *calculated robots*.
- Create a list of *remaining robots* that are to be solved with the following alghorithm.
- Perform depth first search on remaining robots to find two calculated robots for each remaining robot. So for each robot (*k*) in the *remaining robots*
  - Calculate the *degree of linearity* (explained shortly) of all combinations of triangle *k*-P1-P2 where P1 and P2 are selected from the *calculated robots*. *K*, P1 and P2 are variables that can contain a robot identifier. For example P1 can be robot 1. The lower the *degree of linearity,* the better-shaped the triangle, so select P1 and P2 such that the *degree of linearity* is minimised.
    - If P1 and P2 are found: *Solve triangle k*-P1-P2 and remove *k* from *remaining robots* and add *k* to *calculated robots*
    - If not found, put *k* to the end of the list of *remaining robots* and go for the next robot in *remaining robots*
  - If the entire search tree is traversed and no combinations are found, use previous points for all the remaining robots.
- Check for 'flipping', which is when *reference robot* and the *direction robot* cross each other. This is detected by observing other robots since their new location will have a large displacement in comparison with their previous location.
  - If flipping has occurred, recalculate the location of direction robot and perform all the calculations again for this measurement.

Degree of linearity

At the core of the algorithm, positional information is determined using triangulations techniques applied to robot range data. With a flock of robots, many triangles can be constructed. Some of these triangles will be better than others at providing more accurate positional values. In order to select 'good' triangles, a *Degree of linearity* measure is applied. This is used to decide for the best-shaped triangle, which is a triangle that has three equal sides. The Degree of linearity is calculated as follows: Given the three sides, first find the highest side and call it L_1. The others are L_2 and L_3. The Degree of linearity is calculated as $L_1 / (L_2+L_3)$. As this value approaches 1 the shape of the triangle approaches a line. The reason to select the best-shaped triangle is to minimise the error of calculating the actual positions. When a triangle approaches a line any small error in angle calculation can have drastic effects on the final positions. By minimising the Degree of linearity this error is reduced.

This algorithm also deals with uncertainties. If robots are out of range, then the algorithm searches for the best possible alternative triangle. *Solve*

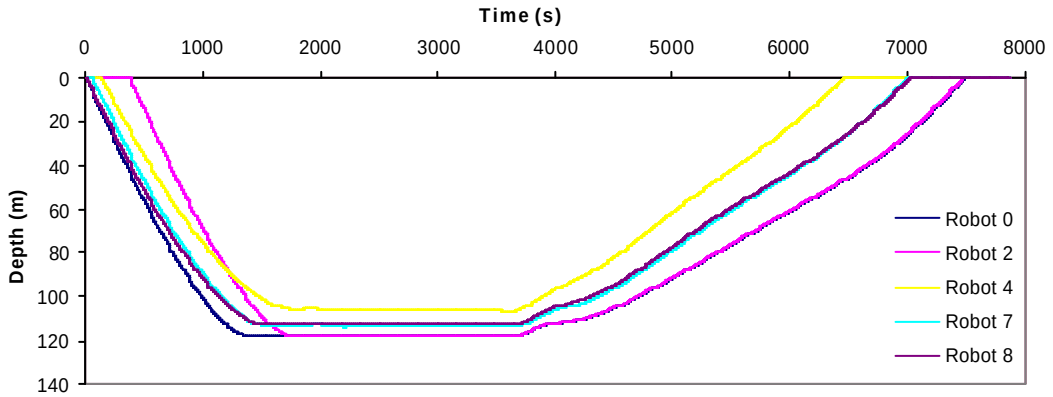

Figure 6: Depth of Divebots over time for some of the robots shown in Figure 5.

triangle is the function that solves the k-P1-P2. This is not as simple as it appears. When solving the triangle there can be multiple solutions and there are also various special cases. For example some sides of the triangle can be zero when two robots happen to be on top of each other vertically. In these cases, the solution is selected by using a heuristic estimation algorithm. Since this algorithm tries to predict the position of the robots when there is no other way to find them, the final result can contain error. For full details on this see (McFarland and Honary 2002).

The result of this algorithm is the relative path of the robots underwater in relation with the two reference robots. One of these is assumed to be in its original location constantly (*reference robot*) and another stays in its original direction in reference with *reference robot*. This robot (*direction robot*) can get closer or further away according to the recorded distance. Therefore the relative result has the same shape of the flock as in reality, except that the flock is rotated and twisted. The results of applying this algorithm to the simulated flock shown in Figure 5 are shown in Figure 7. For simplicity, the seed robots are the same throughout this example. They are robot 0, 1 and 2.

2D/3D velocity profile estimation algorithm

The next step is to calculate the absolute results. This algorithm calculates the absolute position of the three *seed robots*. The term *2D/3D* refers to the fact that there can be two different kind of algorithms, one for 2D and another for 3D. The 2D environment is a vertical slice of the ocean (*x*,*y* dimensions). The robots can go sideways in one dimension and they can go vertically up and down. The 3D algorithm is for the three-dimensional real world (*x*,*y*,*z* dimensions). The algorithm for 2D is

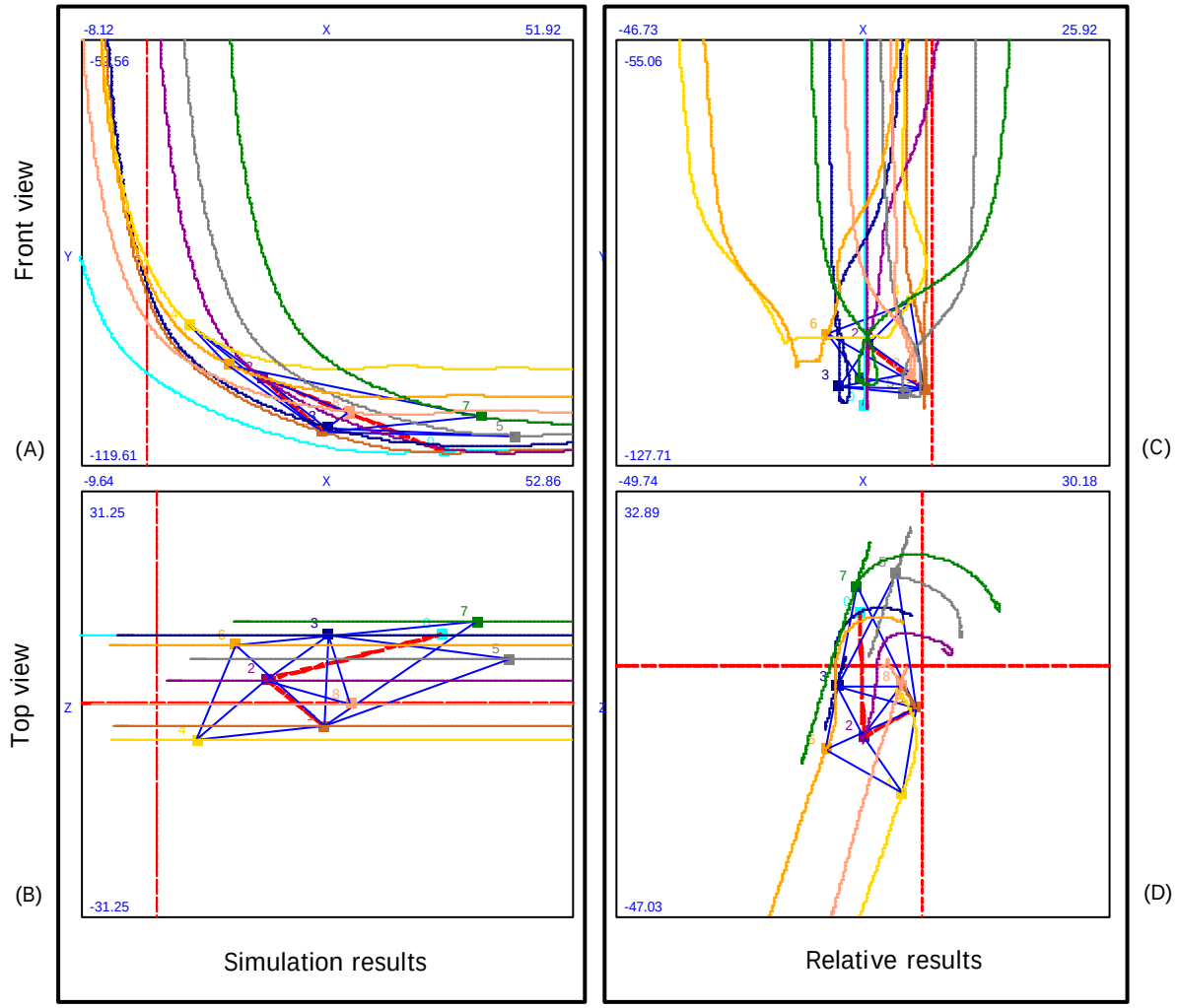

Figure 7: Results of *Relative estimation algorithm* calculated for the example shown in Figure 5. Simulation is shown on the left box (A,B) and the relative results shown on the right box (C,D). For simplicity only two 2D planes (out of 3 shown in Figure 4) are shown for each case. Notice that the coordinates for each 2D plane are different from others. The important point to notice is that the Relative results have the same shape as the simulation results at every measurement. Robot 0 is the reference robot. *Reference robot* is robot 0, *direction robot* is robot 1 and *angle robot* is robot 2.

basically much simpler and hence easier to explain but the same principle is used for 3D. The 2D algorithm is as follows (see Figure 8):

- First calculate the velocity profile for each position using the recorded seed robots, which in 2D are the *reference robot* (R0) and the *direction robot* (R1). Here the *direction robot* is basically the *sampling robot*. This is done iteratively as follows:
  - For each measurement
    - Calculate the velocity profile for R0 using R1. This is achieved by first calculating R1's absolute position using R1's previous location and R0's velocity profile at depth and location of R1. R1 should have experienced the same velocity profile as R0. This is explained shortly.
    - R1's displacement in relation with its previous location (d) contains two elements. The displacement of R0 because of experiencing new current (r_1) and its own displacement because of the current at R1's depth and location (r_2). The second element (r_2) can be obtained from the velocity profile, which is already calculated for the depth and location of R1.
    - Therefore $r_1 = d - r_2$. This is the pure displacement of the reference robot at this new depth and location, which is then used to calculate the current velocity at this point.
    - This value is recorded in the velocity profile for the depth and location of the *reference robot*.
- Calculate the absolute position of all the seed robots based on the calculated velocity profile

In order to reduce the ambiguity associated with a large search space, the algorithm employs certain constraining conditions to make the solution search space tractable. Without these requirements, it is virtually impossible to extract the velocity profile from the relative deformation of seed robots. The requirements for the 2D algorithm are very similar to the 3D algorithm, which will be explained later. The most important requirement to consider at this point is that the seed robots should experience the same velocity profile. This means that R1 will experience the same current if it was at the same depth and roughly the same horizontal location of R0, when R0 was at that depth at an earlier measurement.

In 3D there is a requirement for an extra robot to break the symmetry caused by the increased number of dimensions. This, compared to the 2D case, is called the *angle robot*. The 3D algorithm is as follows: The image in Figure 9 is the simulated movement of the robots from top view in a particular time step, which is at a particular depth. The depth of each robot can be different and they are all known. The illustration is shown in top view (surface of ocean). The image in Figure 10 is the demonstration of the algorithm. For simplicity the

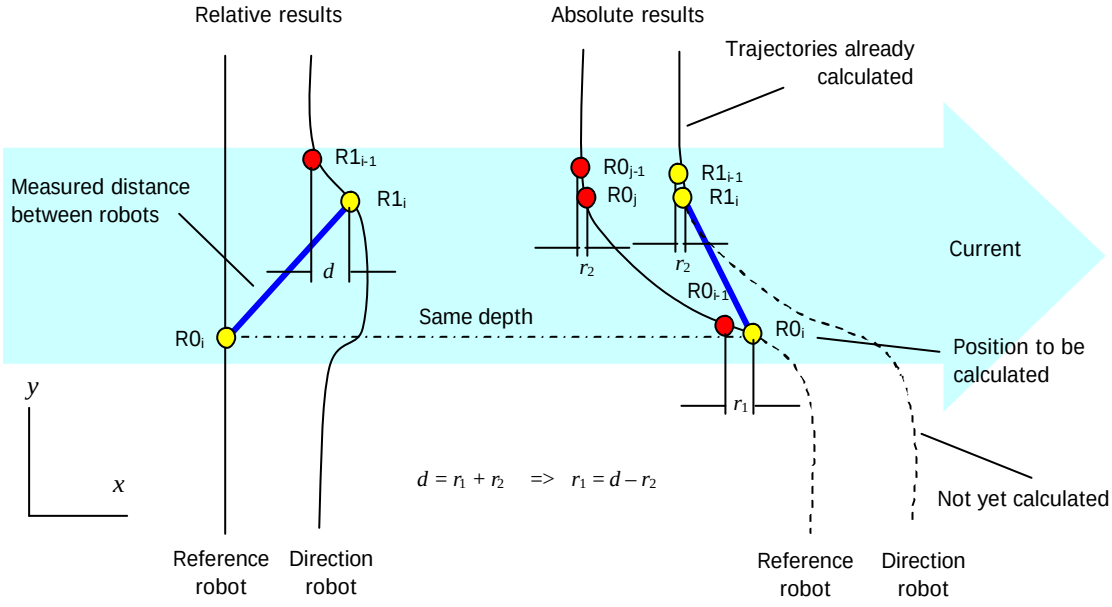

Figure 8: Demonstration of *2D velocity profile estimation algorithm*. This is shown in front view. The left trajectories show the relative results already calculated. The right trajectories show the absolute results to be calculated. *i* is the current measurement and *j* is a previous measurement when R0 was at R1's depth. This is a side view of ocean (down is increasing depth). *Reference robot* is robot 0 and *direction robot* is robot 1.

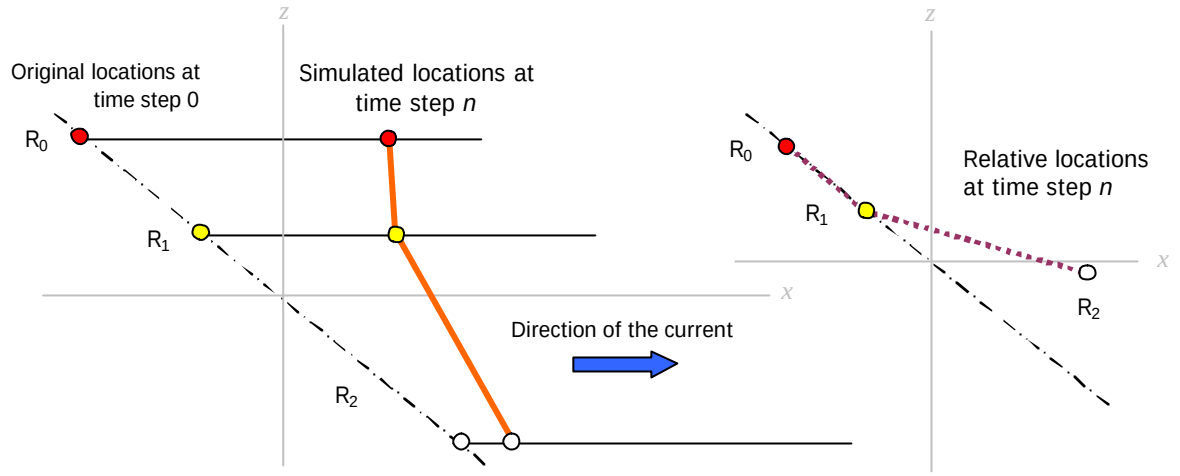

Figure 9: Simulation of current varied at different depth (left). Results from *relative estimation algorithm* (right). The robots have different depth. Top view. *Reference robot* is robot 0, *direction robot* is robot 1 and *angle robot* is robot 2.

three robots are called robot 0, robot 1 and robot 2 for *reference robot*, *direction robot* and *angle robot* respectively. Notice that the shape of the relative flock is the same as the simulated flock. However the position and rotation is incorrect. The aim of this algorithm is to find the absolute position of the seed robots. This is accomplished in the same way as the 2D algorithm. The velocity profile of robot 0 is the actual result. Once that is calculated, the position of seed robots can be calculated. For this the following iterative algorithm is used at this measurement:

1. Obtain N_0, N_1 and N_2, which is the result of *relative estimation algorithm* .
2. Calculate velocity at the depth and location of robot 1 using the velocity profile (V_1). Then calculate $B_1 = PB_1 + V_1$, where PB_1 is the previous absolute position of robot 1. Since robot 0 is always the leader robot (from requirements) and its velocity is calculated as the algorithm goes further, when robot 1 hits the current at a later time, it is possible to use the already calculated velocity to move robot 1 to the right place. Therefore B1 is the absolute position of robot 1 if the velocity profile has been calculated correctly.
3. Calculate velocity at the depth and location of robot 2 using the velocity profile (V_2). Then calculate $B_2 = PB_2 + V_2$ where PB_2 is the previous position of robot 2. This is the absolute position of robot 2.
4. Calculate the translation T to get from N_1 to B_1. Therefore $T = B_1 - N_1$. This is basically trying to match the centre of the flock (N_1) to the new centre (B_1).
5. Transfer N_2 to TN_2 using T. Therefore $TN_2 = N_2 + T$.
6. Transfer N_0 to TN_0 using T. Therefore $TN_2 = N_0 + T$.
7. Calculate the angle *a* between the three points TN2-B1-B2. This will give the amount of rotation needed to rotate TN2 to match B2. After all TN2 should be at B2 if the velocity profile is calculated correctly.
8. Rotate TN_0 around B_1 with *a* so that the shape of the flock stays the same as before. This is now R_0. However this time B_1 and B_2 are at the right place and since the shape is correct, therefore R_0 is the absolute position of robot 0.
9. Using R_0 and the previous position of robot 0 (PR_0), calculate its velocity. This is then recorded in velocity profile for the location and depth of robot 0. The process is then repeated for the next measurement.

Notice that the depth of each robot is always taken directly from the distance table and is combined with the surface results to get the final absolute positions for all seed robots.

The requirements for the 3D algorithm are as follows:

1. The point of origin for *reference robot*, *direction robot* and *angle robot* should be different. Otherwise orientation is lost.
2. *The reference robot* should always be in a *leading* position in relation with the other two robots. This means that when the flock is sinking, the reference robot should be lower than the *direction robot* and the *angle robot*. And similarly when going up the *reference robot* should be higher. This is essential for the calculation of the velocity profile. The choice of seed robots and the reference robot therefore depends on this requirement. This choice is explained in s*eed robots search algorithm*.
3. *Reference robot*, *direction robot* and *angle robot* should experience exactly the same velocity profile, but this can happen at different times. This means that the seed robots should be relatively close together, though not too close to violate the other two requirements.
4. There should be a deformation among the seed robots, so in order to calculate the absolute positions. If no deformation is detected, the previous positions are assumed. Since the

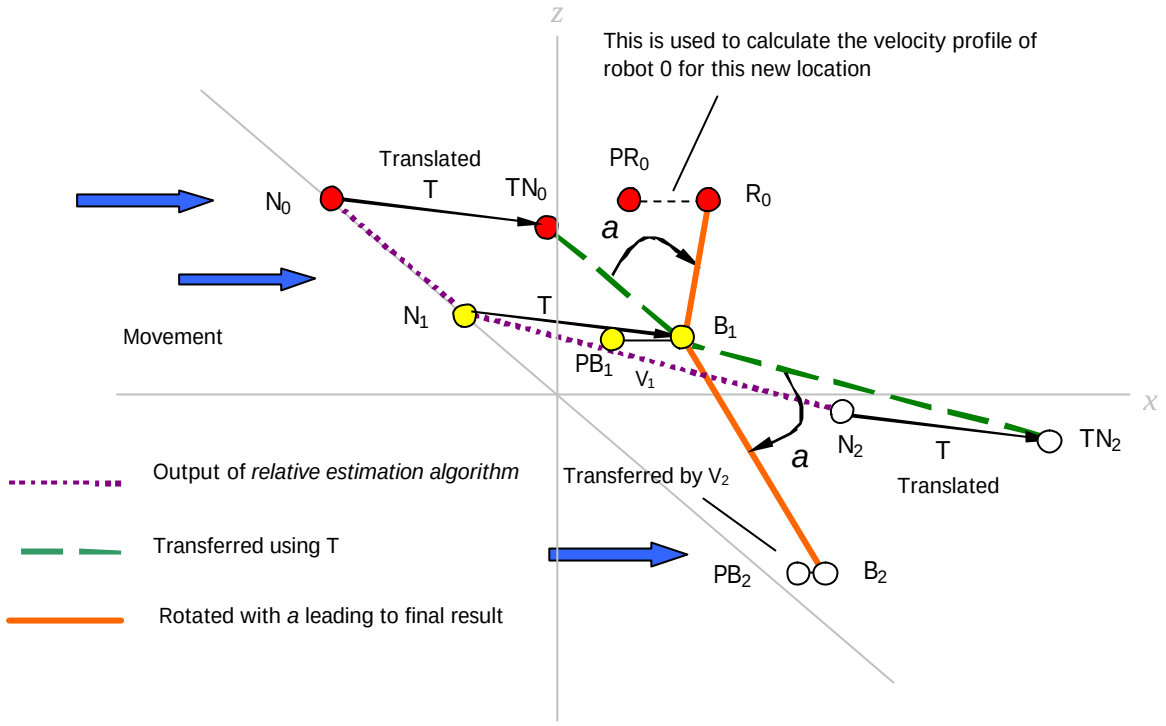

Figure 10: *3D velocity profile estimation algorithm* converts the output of *relative estimation algorithm* to absolute positions. The result (R_0-B_1-B_2) should be similar to the simulation shown in Figure 9. Top view. *Reference robot* is robot 0, *direction robot* is robot 1 and *angle robot* is robot 2.

previous points are not the same as the actual positions, this will lead to calculation error.

Absolute trajectory estimation algorithm

This algorithm is the final part of *flock estimation algorithm*. Once the absolute positions of the seed robots are calculated, the absolute positions of all other robots are calculated by transferring the relative results with the same transformation required to transform the relative seed positions to the absolute seed positions.

In the *relative estimation algorithm* if for any reason (e.g. out of range) some robots are not successfully solved for a particular measurement, they are marked as such. Then at this point an enhancement algorithm is used to find out those positions using linear or spline (such as cubic B-Spline) interpolation. It was found that this technique greatly reduces the amount of error.

Seed robots search algorithm

This algorithm is responsible for deciding for the choice of the seed robots. This can be different at different measurements. This choice is then used throughout the other three parts of *flock estimation algorithm*. Using the distance table, this algorithm tries to find the seed robots at each measurement such that:

- The seed robots are all in range with each other and that their depth are known.
- The seed robots satisfy the requirements set by *3D velocity profile estimation algorithm*.
- Preferably these three seed robots are the closest three of all robots throughout the entire algorithm. This makes it easier for them to have experienced the same current profiles.
- The choice of seed robots during a run is not changed too often. Otherwise extra calculations should be performed every time a new set of seed robots are selected which reduces the performance of the algorithm.
- The reference robot is selected in such a way that it will be the leader when these set of seed robots are selected.

It is also possible to test various seed robots combinations to get better performance out of the estimation algorithm. In this paper the examples are provided with one set of constant seed robots for simplicity. The full realisation of *seed robots search algorithm* is the subject of future research.

Results and Practical Issues

To get the final results, the *2D/3D velocity profile estimation algorithm* and *absolute trajectory estimation algorithm* are applied to the relative results shown in Figure 7. This leads to the absolute trajectories shown in Figure 11. This is basically the final result of the *flock estimation algorithm* applied to the original *distance table*. Therefore the final results should look very similar to the original

simulated trajectories shown in Figure 5. The shape of the flock is quite similar to the simulation. There is only a small amount of error. In order to compare the simulation and estimation the following performance measure is used. Let x_1, y_1, z_1 be a simulation point at measurement i for each robot and x_2, y_2, z_2 be the corresponding point for absolute estimated results. The distance between these two points is calculated as

$$d_{r_i} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

The error over measurements (e_i) is then calculated as the average of this distance for all the robots at a particular measurement (i) scaled for the length of the trajectories of all robots. Therefore:

$$e_i = \frac{\sum_{r=0}^{R} \frac{d_{r_i}}{L_r}}{R}$$

where R is the total number of robots and L_r is the length of the trajectory of robot r. Since there are finite number of measurements, L_r is calculated as the total of all distances traveled from each measurement to the next as an approximation of the total length. As a result e_i shows a percentage of error depending on the distance the robots have traveled. This way e_i for different experiments can be compared together. The e_i for this example is shown in Figure 12. The positional error increases along the trajectory. If at any point there is an error, this error will be accumulated over time. This is the direct result of both relative and velocity profile algorithms. The error in this example is mainly created as a result of special cases when calculating the relative results. For example, special cases can occur when the shape of the flock is symmetrical and that calculating the position of a robot leads to two solutions where it is not easy to select the right answer easily. This situation does not happen often, however once it happens it will create an error, which is then accumulated. For a detailed description of special cases see McFarland and Honary (2002).

Classification of errors

There are a number of mechanisms which can lead to error in the final trajectory. These errors can be the result of uncertainty in the acquired measurements, which will then lead to position errors in the estimated results. In reality any measurement contains an associated error. To simulate this, a random value (with Gaussian

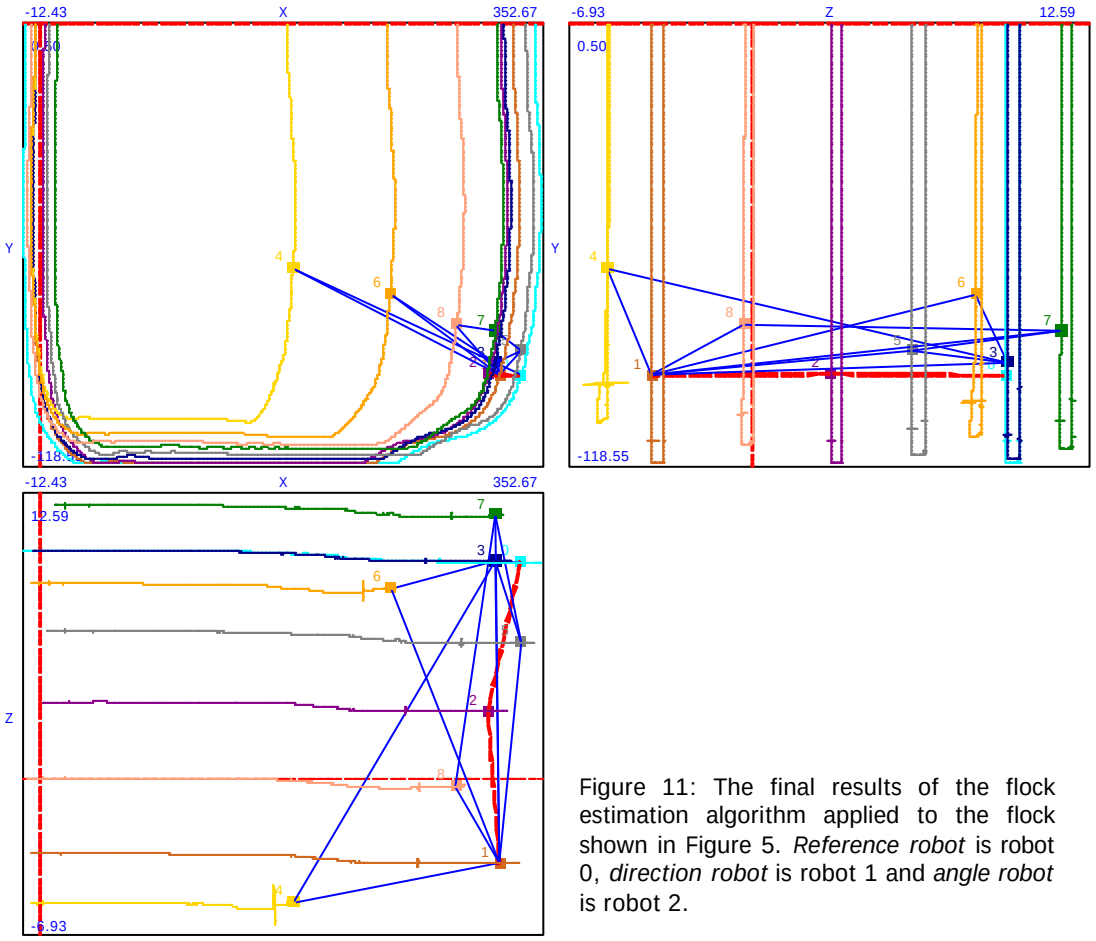

Figure 11: The final results of the flock estimation algorithm applied to the flock shown in Figure 5. *Reference robot* is robot 0, *direction robot* is robot 1 and *angle robot* is robot 2.

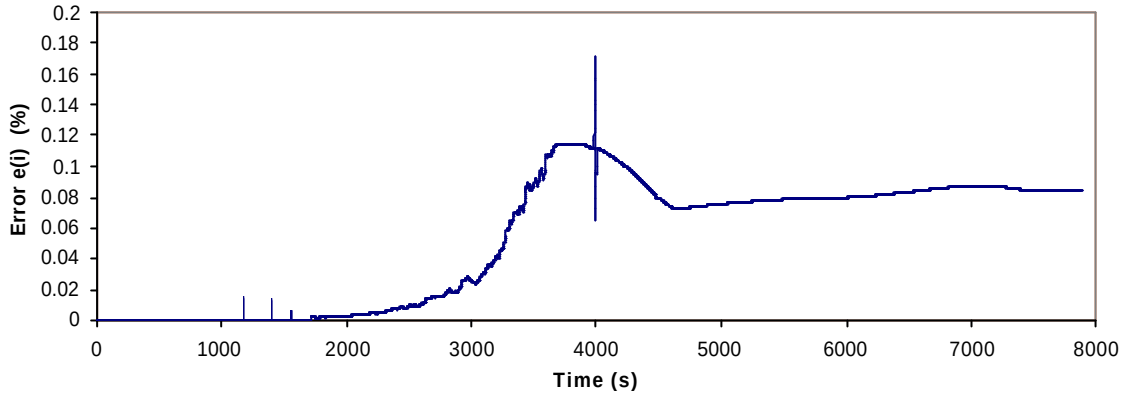

Figure 12: The error e_i shown for all the measurements for simulation shown in Figure 5. The error is the average percentage of error over the length of the trajectory of each robot. he spike is due to special cases that lead to error in calculation of seed robots themselves. As a result all other robots are slightly displaced, which affects the overall error at that measurement.

distribution) is added to the recorded distances before they are recorded to the distance table. This tests the performance envelope of the algorithm. What follows is the classification of certain problems that lead to error in the final results. Ways to overcome some of these issues are discussed in the next section.

- *Seed robots*. The most important error comes from seed robots that have different specifications. If seed robots experience different currents due to different behaviour, or different manufacturing characteristics or different currents experienced then they will produce error, which will be magnified. Basically if the main requirements are not fully satisfied the estimated seed robots would have error in their calculated trajectories.
- *Number of measurements*. Generally more measurements lead to better accuracy of the results as there will be more information. However, since the flock distortion algorithm processes the measurements sequentially and since any error can accumulate as the algorithm proceeds, the more measurements there are, the more accumulation of error there will be. On the other hand the less measurements there are, the less deformation is recorded since the resolution is not high enough to detect fine deformations. This leads to error as the estimated trajectories would have a low resolution. So somewhere in between is the optimized solution. This is in fact subject of future research.
- *Range*. The algorithm tries to find the best solution when robots are out of range. However, if a robot is completely out of range of all other robots, its position is guessed and therefore it will contain error. This error is usually projected into the relative results first.
- *Inaccurate distance measurement and original locations.* This is an important error as the entire calculation is heavily based on distances. It is also known that underwater distance measurements, usually performed by sonar, are subject to error and this should be seriously considered in the algorithm. An example for this kind of error is shown in Figure 13. Here the recorded distances are subjected to Gauissian noise with s = 0.001 of the length of each link. The e_i is shown in Figure 14.
- *Flock flow*. The way the flock is distorted has an effect on the performance of the seed robots. For example, if seed robots are further away from each other, they will be deformed more and this leads to better results as the triangulation is performed with larger triangles. On the other hand the further the seed robots are from each other, the more likely they will experience a different set of currents which will then lead to a different kind of error.
- *Special cases*. These cases in *relative estimation algorithm* always lead to errors when the profile is calculated later on.
- *Error cascade*. This happens depending on the shape of the flock. Certain configurations can lead to a situation where the solution of a robot (R3) depends on two other robots (R0 and R1) and then the solution of another robot (R4) depends on R3 and R1 and so on and so forth. This is a sequential error cascade. If the first robot has any error (R1), this cascade will magnify the error enormously for the robot in the end of the cascade (R4). In this kind of situation the further away a robot is from the centre of the flock (seed robots) the more error it has. Also the further away a robot is from other robots, the longer the sides of the

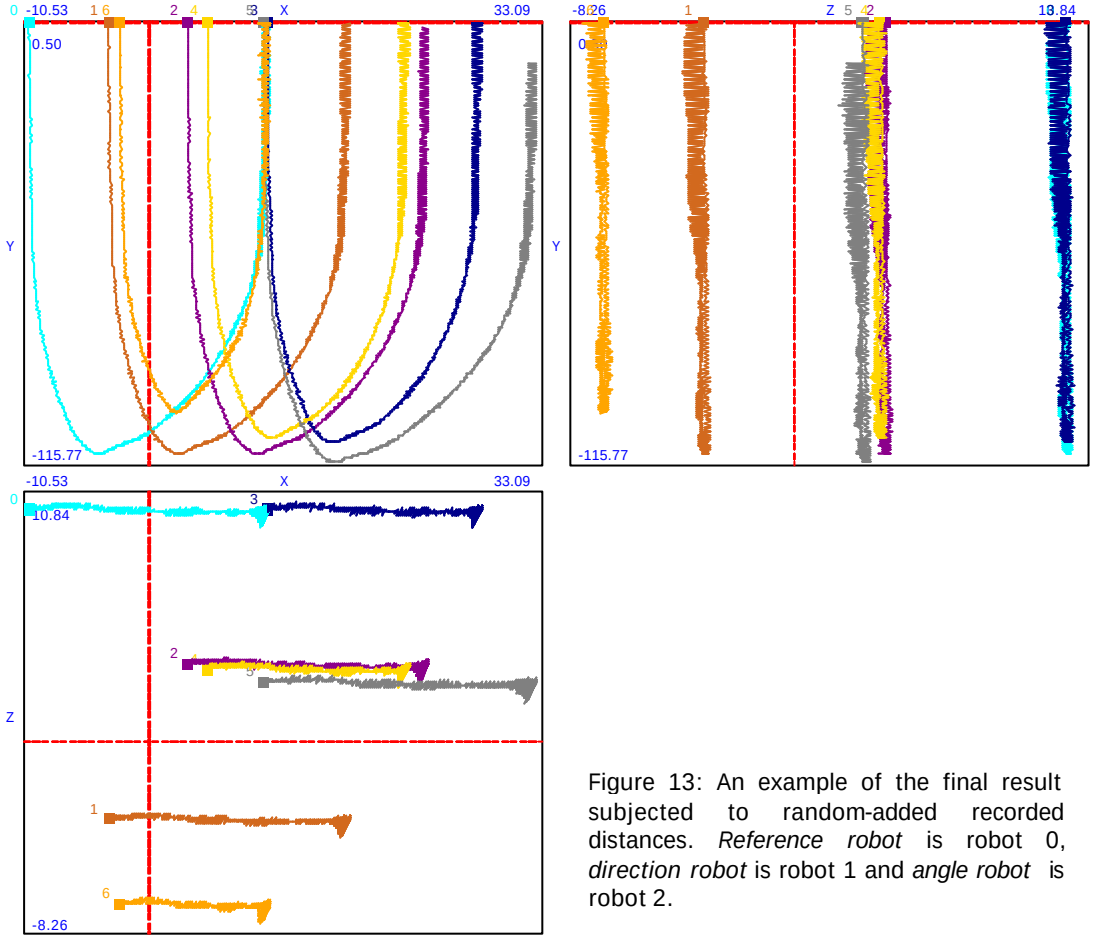

Figure 13: An example of the final result subjected to random-added recorded distances. *Reference robot* is robot 0, *direction robot* is robot 1 and *angle robot* is robot 2.

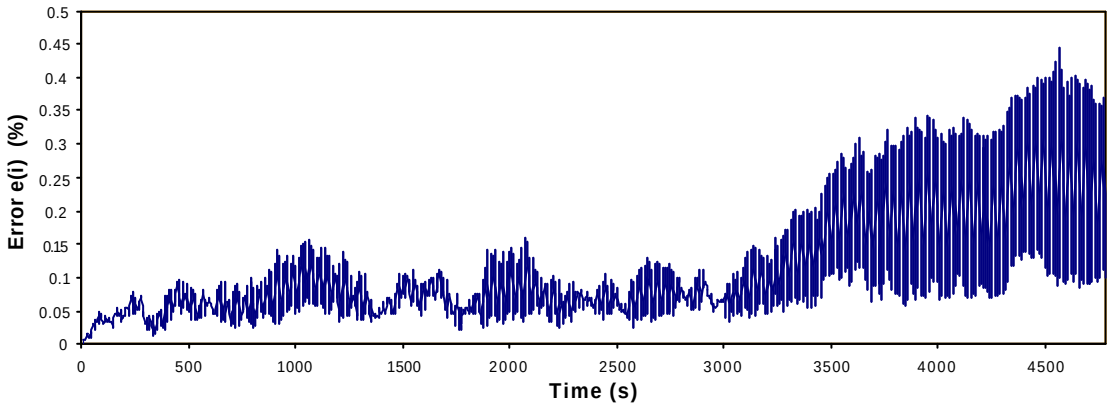

Figure 14: The error e_i for example shown in Figure 13.

triangle. This can lead to more error as illustrated in Figure 15.

- *Flock distortion*. Error created as a result of deformation of the flock. If the flock is not deformed, the error stays constant. Also if the resolution of the sampling is large in such a way that the deformation of the seed robots happens between this time interval, then no deformation is recorded and so no results can be produced. Therefore it is important to keep the time interval between samplings small enough so that the seed robots can be deformed.
- *Classes of currents*. The more complicated the classes of currents get, the more error there will be. This however depends on many parameters such as deployment shape, timings, resolution of sampling and so on.

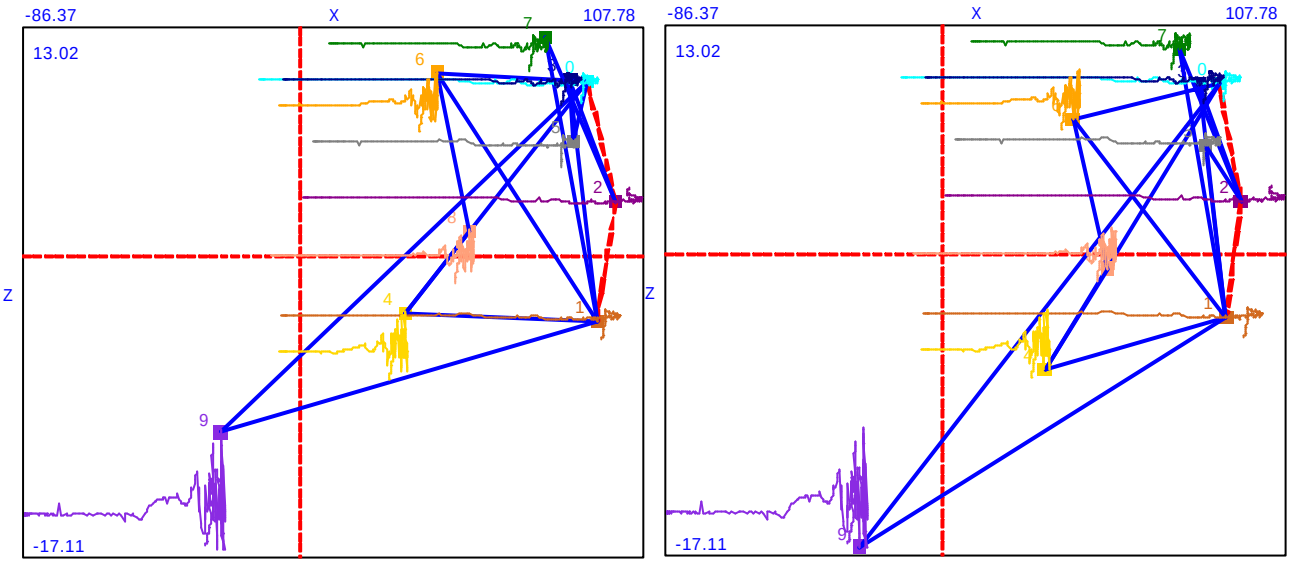

Figure 15: The cascade error. The left (A) and right (B) figures correspond to two different measurements. Both (A) and (B) are top view of the estimated absolute trajectories. Notice that the shape of the flock stays the same, though it is rotated as a result of small error in the beginning of the cascade. *Reference robot* is robot 0, *direction robot* is robot 1 and *angle robot* is robot 2.

Enhancing the algorithm and future work

To overcome the different types of errors explained in the last section, a few enhancing methods are considered. A handful of these methods are presented here to show that it will eventually be possible to get reasonable results in real-world applications. Reducing the error is the subject of future research.

- *Enhancing seed robot search algorithm.* This algorithm will try to search for the best combination of seed robots. These are theoretically identical robots that have satisfied the requirements. In practice these robots are selected in such a way to have the most common specifications. The choice of seed robots can change in between the measurements. This means the relative and then absolute results are first calculated for the first set of seed robots and then the whole algorithm is applied again as if the last absolute results are the original locations for the next series and the calculation is performed once again. Also it is better to select the seed robots to be at the center of the flock to minimize the *cascade error.*
- *Choosing the sampling rate.* The resolution can be set in two places. One is in the actual robots were they conduct the physical sampling. The other is in the program, which can reduce the sampling resolution by skipping some of the sampled data. For the purpose of this algorithm the second method is enough to experiment with and get various results. The algorithm can be applied to the distance table with various sampling rates to find out the best-optimised sampling rate.
- *Cross scanning*. The current algorithm only uses the original formation of the flock as the starting point. It is possible to use the final formation (GPS locations when the robots resurface) and apply the algorithm backwards to the distance table. The two results can then be average or statistically combined to reduce possible error.
- *Optimising deployment formation.* The deployment formation is a very good way to minimize error. A set of robots can be assumed for the functionality of seed robots. These are dropped close to each and perhaps at the center of the flock. It is also possible to have multiple sets of seed robots at different parts of the flock in case some of these seed robots are lost. Later the algorithm can calculate the results for each set of seed robots and then combine or average all of them to get a better solution. This is also the subject of future research.

Another important aspect of the flock distortion is the flocking behaviour of the group of robots. Since they only have one dimension of freedom and very limited knowledge about the state of other robots, a minimal approach should be taken. The implementation if this minimal algorithm is the subject of future research.

Conclusion

In this paper a novel technique was demonstrated to use the deformation of a flock of robots to obtain the absolute 3D path of the flock moving through an environment. The environment used in this paper was underwater, though it can be other environments such as an atmosphere. Each robot in the flock only records the distances between itself and other members of the flock (or as many as are in range) and uses these values as well as the original and final locations of the members to obtain the absolute trajectories. Therefore, measuring the bearing between the robots is avoided. This reduces the complexity of the robots design and hence leads to reduced costs of manufacturing. Since there are many robots in the environment working as a flock it is possible to collect more data in a given time as the robots are working in parallel and the area under consideration is sampled with higher precision. Redundancy is also increased as there are more robots. If for any reason some of the robots are lost or their data is not successfully collected, the algorithm can still use other robots to calculate the environmental variables. The benefit of this approach is also to extend the range of the operation (or depth) as there is no need for marker communication systems. For example, the flock can operate under surface ice where it is not possible to get a GPS fix due to inability of resurfacing.

The flock distortion algorithm can also take advantage of multiple robots to find a better solution in comparison with using individual robots. For example it is possible to use statistical averaging to enhance the final results to remove any possible inconsistencies from the collected measurement data. There are limitations and advantages on the number of robots that can be used with this flock distortion. This topic is discussed thoroughly in McFarland and Honary (2002). In general the advantage of using a flock distortion technique is that it can potentially provide higher resolution sampling data for the same number of robots used in comparison with conventional Lagrangian methods.

References

Clarke, M.R., (1979) *The head of the sperm whale*, Scientific American, 1979, vol 240, p106-117.

Davis, R.E., Webb, D.C., Regier, L.A. and Dufour, J., (1991) *The Autonomous Lagrangian Circulation Explorer (ALACE)*, J. Atmospheric & Oceanic Technology, 9(3), 264-285.

Davis, R.E., (1991) *Observing the general circulation with floats.* Deep-Sea Res., 38, Suppl. 1, S531-S571.

Roemmich, D., et al. (1999) *On the design and implementation of Argo: A global array of profiling floats*. (White papers from the Argo science team)

McFarland, D. and Honary, E. (2002) *Flock Distortion: A new approach in mapping environmental variables in deep water.* Robotica (in press)

McFarland, D., Gilhespy, I. and Honary E. (2002) *DIVEBOT: A diving robot with a whale-like buoyancy mechanism*. Robotica (in press)

Swallow, J.C., McCartney B.S. and Millard, N.W. (1974) *The Minimode float tracking system,* Deep-Sea Research, Vol 21, pp 573-595.