

# Using CFD in Robotic Simulators for pollution monitoring

John Oyekan, Bowen Lu, Huosheng Hu and Dongbing Gu,  
 School of Computer Science and Electronic Engineering,  
 University of Essex, Wivenhoe Park, Colchester CO3 4SQ, United Kingdom.  
*Email: jooyek;blv;hhu;dgu; @essex.ac.uk*

**Abstract**—This paper presents the RoboShoalSim, a robot simulator that was developed for the EU Robotic fish shoal project to monitor pollution levels in sea ports around the world. The simulator is different from those in use in that it incorporates not only pollution models of the pollutant in the simulated port but uses computational fluid dynamics of the sea port to generate closer to reality pollutant dynamics. In addition, a physic engine is used with the aim of studying the effects of the Robotic fishes dynamics on the pollutant plume structure. In this paper, the general architecture is described, with preliminary examples of utilization.

**Index Terms**—UAV, Bacterium Inspired Algorithm, Environmental Monitoring, Flocking, Computational Fluid Dynamics.

## I. INTRODUCTION

Monitoring the quality of water in ports around the world is fast becoming a priority to various governments. A good example is the EU Shoal Project that aims to monitor the levels of pollutants in water and develop a 3 dimensional map of the pollutant by using a shoal of robotic fishes. In addition, these robotic fishes would be used to find the source of leaking pipes or vessels intentionally discharging pollution into natural water bodies.

This project involves the development of advanced swarm intelligence for the robotic fish to be used and also investigates how the hydrodynamics of the environment in which the robotic fishes are deployed affect the efficiency of the developed swarm intelligence algorithm. As it is virtually impossible to develop and test algorithms on the physical shoal of robotic fishes, it is advisable to use a robotic simulator first before real life physical deployment.

This approach has been followed by various researchers investigating the development of source seeking controllers for physical underwater robots. Some have developed custom based simulators while others have used commercially available simulators. For example Tu et al developed a custom simulator in order to investigate fish behaviours in [1] while in [2], generating of fish motion for use on a physical platform was investigated. In [3], Nawaz et al developed a simulator to test a sensor monitoring system that was to be deployed to monitor underwater nuclear storage pools. However, these scenarios assumed an environment

with little or no disturbance or turbulence affecting the plume or pollutant concentration generated by a pollutant source. In order to introduce disturbance into the environment, one might consider using a generated flow field and a random number generator but this does take into consideration the structure of the environment. In order to take the structure of the environment into consideration, computational fluid dynamics of the environment needs to be performed. However, computational fluid dynamics simulations take up a lot of computational resources and it is not possible to perform in real time especially if the environment to be simulated is very large and the grid resolution is very small. Farrell et al [4] were able to solve this problem partially by using a set of computational fluid dynamics equations with a colored noise process but without taking the structure of the environment into consideration. This led to a plume that meandered in the flow field but did not interact with the physical environment. However in [5], Zarzhitsky et al used Farrell et al's code in an environment with obstacles and this caused resulted in plume interaction.

In this work, we develop a 3 Dimensional simulator that takes into consideration the structure of the environment whilst generating a plume that interacts with the environment structure. Various simulators that could be used in this work were investigated prior to development. For example, the webot simulator was considered. The webot simulator is a 3 D commercial simulator that has been around for over 15 years [9]. It includes robot models for the pioneer robot [6], sony aibo [7], kephera robots [8] and so on. However, we considered not using this because of its proprietary nature. In addition, access to the underlying code to modify it to suit the Robotic Shoal project is restricted and hence another discouragement from using webots. Another simulation environment that was considered was MobileSim by ARIA. This simulation environment did not have models for the robotic fish and modifying the underlying development code was considered to be a lot of work especially in the time scale of the project. Simbad 3D simulator developed in Java was also considered [10]. In all the simulators mentioned so far and considered, it was not clear how to the pollutant particles that make up the plume structure of pollution could be developed

readily hence another reason why building a custom simulator was considered. By building a custom simulator for the Robotic Shoal project, it was possible to have full control over the development process in various ways including the introduction of various computational fluid dynamics flow fields, sensor models, custom built robots and so on.

As a result of the above limitations, Panda3D [11] was considered has a suitable platform. The platform has the following features:

- It is an open source game engine developed in Python scripting language which has a lot of support in the research community. This made it possible to use readily available codes to aid the simulator development. Python could also be written in an object oriented way which makes it suitable to use when multiple objects need to be developed. Panda3D also has a C++ extension that can be used to develop programs.
- It has a physics engine that enables collision between objects to be detected and addressed accordingly.
- Recently, an artificial intelligence module that comprises of behaviours such as flocking, obstacle avoidance, pursue etc as been added. These behaviours could be used on simulated robots when building a simulator.
- The use of textures on objects in Panda3D is very easy and so is camera control. It also offers integration with model development software such as Blender, 3dMax and so. In other words, the use of Panda3D encouraged flexibility that was needed in this project to achieve different things.

The rest of this paper is organised as follows:

## II. ARCHITECTURE AND FRAMEWORK

In addition to using Panda3D, a variety of other tools were also used in order to compliment each other. The framework is shown in Figure 1 An explanation of the various tools and how they were used will now follow.

- In order to develop the simulator, a picture of the environment as shown in Figure 2(a) was first taken. This picture was then used in Blender and OpenFoam computational fluid dynamics package.
- Blender is an open source development package that can be used to develop 3 D worlds and introduce various effects [12]. Blender also incorporates a physics game engine and particle system which makes it possible to introduce interactions between various objects in the simulated environment and also create smoke, moving water and other similar effects. However, it was discovered that there was no control over each individual particle in a user developed particle system. As a result, it was impossible to control the shape of a simulated pollutant plume. Blender was used in this project to develop the environment of interest by following the picture taken as shown in Figure 2(b) and to develop

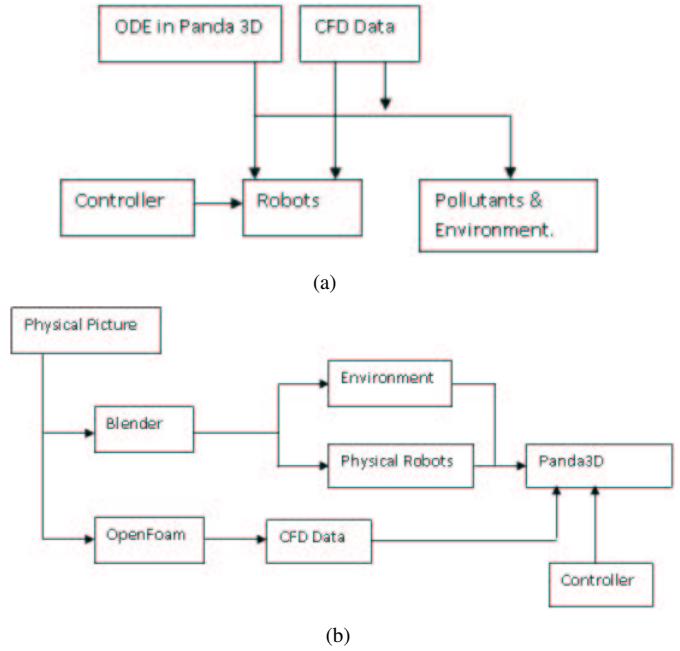
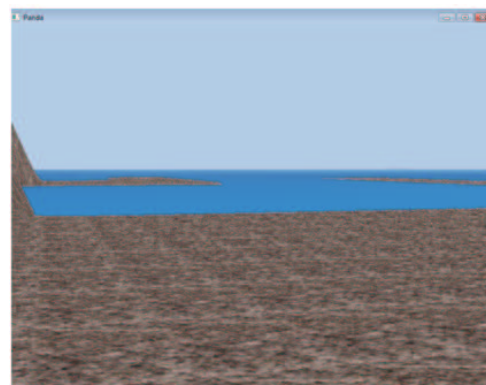


Fig. 1. Developing RoboShoalSim. Fig. (a) shows the framework used; Fig. (b) shows the steps followed towards construction.

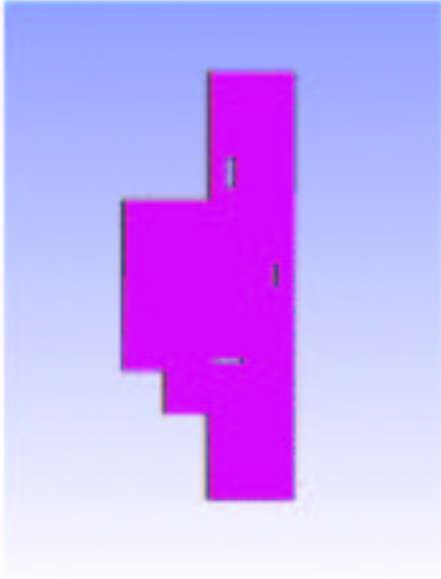


(a)

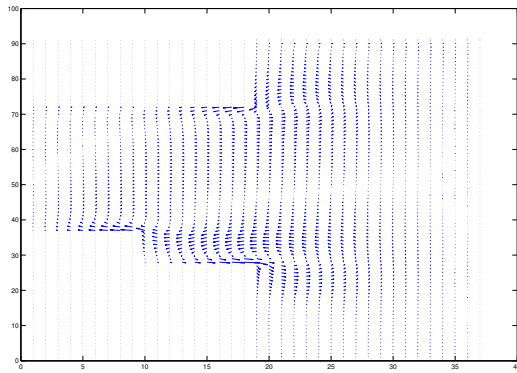


(b)

Fig. 2. Showing the physical environment- Fig. (a) and the simulated environment- Fig. (b)



(a)



(b)

Fig. 3. Showing the meshed simulated environment- Fig. (a) and showing the velocity vectors obtained from openFoam in the simulated marine environment- Fig. (b)

the robotic fish. The environment developed was setup in Blender so that when imported into Panda3D, the boundaries of the environment could act as actual physical boundaries to prevent objects from going through them. This was achieved by using the chicken tool in Blender. In developing the robotic fish, a bone frame based on a 3D fish model was used as shown in Figure 4. In this design, each of the joint is customisable, including quantity and organisation of bones. Although we are using point model for fish movement at current stage, this bone frame can potentially support kinematic methods in future work.

Figure 5 shows textured fish model. As mentioned

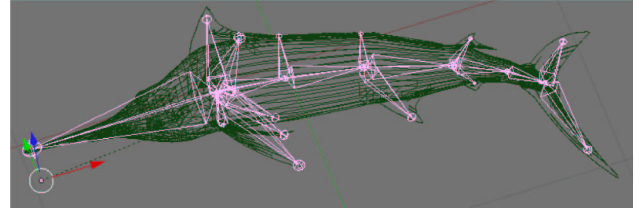


Fig. 4. Bone frame of fish model

above, we created a series of animation of swimming fish in different flipping frequency for different moving speed. The robotic fish were deployed as shown in Figure 6.

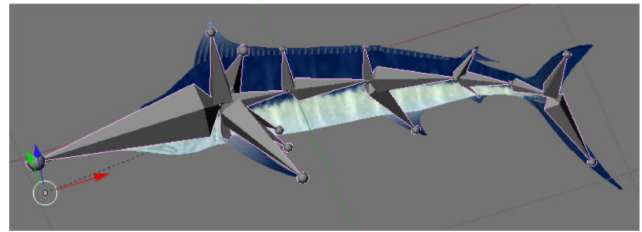


Fig. 5. Textured fish model

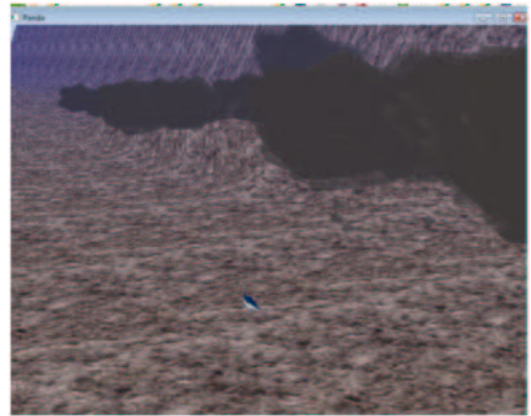


Fig. 6. Simulated robotic fish deployed in simulator

- OpenFoam [13] is an open source computational fluid dynamics program that incorporates its own mesh generators e.g snappyHex and blockMesh. The process of meshing simply converts an environment into grids or cells that enable the computer to perform finite element computations across the environment of interest. The environment shown in Figure 2(a) was simplified and converted into a mesh as shown in Figure 3(a). In order to perform Computational Fluid dynamics, equations that govern the flow of fluid in the environment have to be derived. The choice of these equations would depend on whether the fluid is to be turbulent, laminar,

viscous, incompressible and so on. After the equation to describe the fluid as been chosen, the environment is divided up discretely using a mesh generator so that the fluid equation could be solved discretely for the various areas in the environment [13]. Depending on the size of the environment and the resolution of the meshes, the results from the computational fluid simulation could take up to days. As a result, it is very challenging to perform real time computational fluid dynamics of an environment without incurring computational overheads. In order to obtain computational fluid dynamics results quicker, the environment to be simulated was reduced by a factor of 10 before expanding back into normal real life dimensions. The results of doing a computational fluid dynamics computation on the Figure 3(a) is shown in Figure 3(b).

The structure of OpenFoam enables users to easily establish different flow regimes ranging from laminar to turbulent by setting flow parameters in equations. For more information, the reader is referred to the OpenFoam manual at [13]. OpenFoam was used to generate flow field data which was then used in Panda3D to advect pollution particles and affect the dynamics of the Robots in the simulator.

- With the framework shown in Figure 1, it was possible to make a module that could incorporate various controllers for testing on physical robots in Panda3D.

### III. SIMULATING POLLUTION

Simulating pollution was performed by using a smoke structure developed in Blender and shown in Figure 2(b). This smoke structure was not set to be a boundary because it was assumed that each smoke structure was made up of a number of very small particles. Robot interaction with these particles do not affect the structure of the smoke. Each smoke structure was advected by the velocity flow field and by emitting the smoke structures every time step, a plume is generated as seen in Figure 2(b). The lifetime of each smoke structure can be set so that they die and can be "reborn" at the emitter location. This makes sure that memory is not used up with more and more smoke structures being born and also makes sure that the simulator does not suffer from many idle objects in memory. In order to simulate dispersion, the smoke structure is slowly increased in size by using panda3D's setScale function and the transparency factor increased every time step by using the function setTransparency. In order to achieve meandering, a random number generator is used with the flow field generated by the OpenFoam package for each smoke structure at every time step.

### IV. A BIO-INSPIRED CONTROLLER

In order to test the frame work of RoboShoalSim, we used a bio-inspired controller that is based on the bacteria chemotaxis behaviour and the flocking behaviour of starlings.

This controller uses the bacteria chemotaxis behaviour to navigate the Robotic fish towards the source of the pollution whilst the flocking controller is used so that collective group foraging is possible. This controllers will now be briefly described below.

#### A. Bacteria Controller

The Bacteria foraging behaviour is made up of a combination of runs and tumbles. A tumble motion is one in which the bacteria spins into a random direction while a run motion is essentially a straight motion. When a bacteria is moving in a favorable direction, the frequency of tumble behaviours are reduced and vice versa when moving in an unfavorable direction. This behaviour makes the bacteria navigate towards rich food sources and the randomness in its motion equips the bacteria with a rich exploratory behaviour. This enables it to escape from local minimas in its environment. In this work, the Berg and Brown controller used by Oyekan et al in [14] is used to control the Robotic agents. This model was chosen because of its ease of tuning and the possibility to compare artificial agents performance to that of their biological counterparts. This model is showed in equations 1 to 3. Where  $\tau$  is the mean run time and  $\tau_o$  is the mean run time in the absence of concentration gradients,  $\alpha$  is a constant of the system based on the chemotaxis sensitivity factor of the bacteria,  $P_b$  is the fraction of the receptor bound at concentration  $C$ . In this work,  $C$  was the present reading taken by the Robotic agent.  $K_D$  is the dissociation constant of the the bacterial chemoreceptor.  $\frac{dP_b}{dt}$  is the rate of change of  $P_b$ .  $\overline{\frac{dP_b}{dt}}$  is the weighted rate of change of  $P_b$ , while  $\tau_m$  is the time constant of the bacterial system. The above Equations determine the time between tumbles and hence the length of runs between tumbles. During the tumble phase, the agent can randomly choose a range of angles in the set  $\sigma \in \{0, \dots, 360\}$ . The range of the angle made it possible for the robotic agents to backtrack if there was a favourable gradient behind them.

$$\tau = \tau_o \exp\left(\alpha \overline{\frac{dP_b}{dt}}\right) \quad (1)$$

$$\overline{\frac{dP_b}{dt}} = \tau_m^{-1} \int_{-\infty}^t \frac{dP_b}{dt'} \exp\left(-\frac{(t' - t)}{\tau_m}\right) dt', \quad (2)$$

$$\frac{dP_b}{dt} = \frac{k_D}{(k_D + C)^2} \frac{dC}{dt} \quad (3)$$

In order to make sure that the robotic fish does not overshoot its target, a velocity function as shown in equation 4 was used in conjunction with bacteria controller. where  $\beta$  is the dynamic velocity that depends on the present reading of the environmental quantity  $C$ ,  $\beta_o$  is the standard velocity without any reading and  $v_k$  is a constant for tuning the dynamic velocity  $\beta$ .

$$\beta = \frac{\beta_o * v_k}{C} \quad (4)$$

## B. Flocking Behaviour

The flocking behaviour is one that is used by living organisms such as starlings. The behaviour enables collective foraging also aids the group's success in finding a food source and exploring the environment. Craig Reynolds was the first to model this behaviour in his work on boids in 1986 [15]. His model has gone on to be used by various researchers using various models [15]. In this work however, the morse potential as shown in equation 5 is used instead. where  $r$  is the Euclidean distance between two closest neighbours. Gains of 1 for the repulsion term  $G_R$  and 0.99 for the attractant term  $G_A$  were used. It was discovered that it is possible to control how closely the agents get to each other whilst not colliding by adjusting the  $G_G$  gain.

$$F_{output} = G_G * [G_R * exp(-r/20) - G_A * exp(-r/20)] \quad (5)$$

## V. PRELIMINARY TESTS

In order to test the feasibility of the proposed structure and particularly the use of Panda3D as a platform to develop the simulator, experiments were carried out to see if simulated Robotic UAVs could be used to form a visual representation of an invisible pollutant. This is shown in Figure 7. In this experiment, the pollution sensor reading at a position  $(x,y,z)$  was obtained by using the equation 6.

$$PollutionReading = \sum_{i=0}^m \frac{Q_i}{R_i^2} \quad (6)$$

A chemical cloud was simulated by using particles. Each particle making up the chemical cloud was given a  $Q$  of 10000.  $m$  is the number of particles in the environment and  $R$  is the distance of the agent from the position of a particle in the environment. It is assumed that the chemical sensors used in the experiment were noiseless during the simulation. We plan to introduce noise through the use of a random number generator in subsequent experiments to investigate its effect on the algorithm. It is also assumed that there were no background noises affecting the quality of the data collected by the robotic agents. The chemical cloud particles were subjected to a diffusion type brownian motion about the centre position of the chemical cloud. By using the setFluidPos() function of Panda3D, it was possible to make the particles move fluid like whilst being advected and also take collisions with other objects into consideration. Advection of the chemical cloud and hence the particles was simulated by subjecting the centre point to a constant velocity of  $(0, 2.0, 0)m/s$  with UAV agents having a top speed of  $2m/s$ .

The results shown in Figure 8 show that the agents were able to form a visual representation of the distribution of the chemical cloud. These preliminary results also show that the framework proposed in section II is capable to be use in RoboShoalSim.

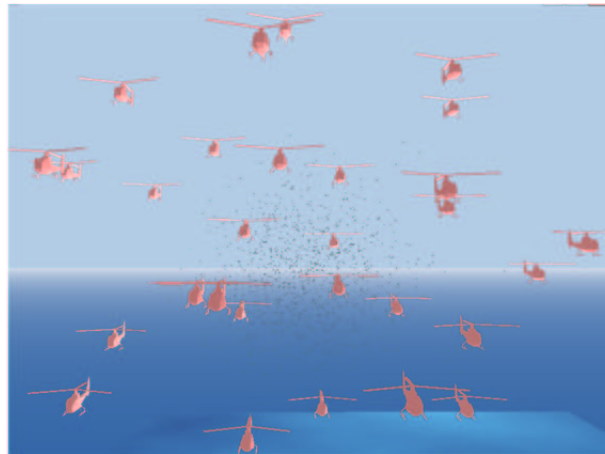


Fig. 7. Showing the 3 Dimensional simulation Environment.

## VI. CONCLUSION

A simulator framework capable to be used to test algorithms to be deployed on Robotic fishes in the EU Shoal project has been proposed in this paper. Preliminary tests confirm this even though it was performed in a different environment. With the framework, it is possible to use Panda3D's open source nature to introduce algorithms and features that would enable the testing of various pollution seeking algorithms. According to present knowledge, this is the first time that Panda3D has been used with computational fluid dynamics data and to develop a simulator for a robotic fish. The bacteria-flocking controller used in this paper has a down side in that it is known to fail in very turbulent environments. Future work would investigate the use of more robust algorithms and might also consider using the Farrell et al approach [4] in a 3 dimensional environment

## ACKNOWLEDGEMENTS

This research work is financially sponsored by European Union FP7 program, ICT-231646, SHOAL.

## REFERENCES

- [1] D. Terzopoulos, X. Tu and R. Grzeszczu, *Artificial Fishes: Autonomous Locomotion, Perception, Behavior, and Learning in a Simulated Physical World*, *Artificial Life*, 1(4), pp 327351, 1994.
- [2] J. Liu and H. Hu., *Biological Inspiration: From Carangiform Fish to Multi-joint Robotic Fish*, *Journal of Bionic Engineering*, Vol. 7, No. 1, pp 35-48, March 2010.
- [3] S. Nawaz, M. Hussain, S. Watson, N. Trigoni and P.N. Green, *An Underwater Robotic Network for Monitoring Nuclear Waste Storage Pools*, *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, Vol. 24, pp 236-255, DOI: 10.1007/978-3-642-11528-8\_17, 2010.

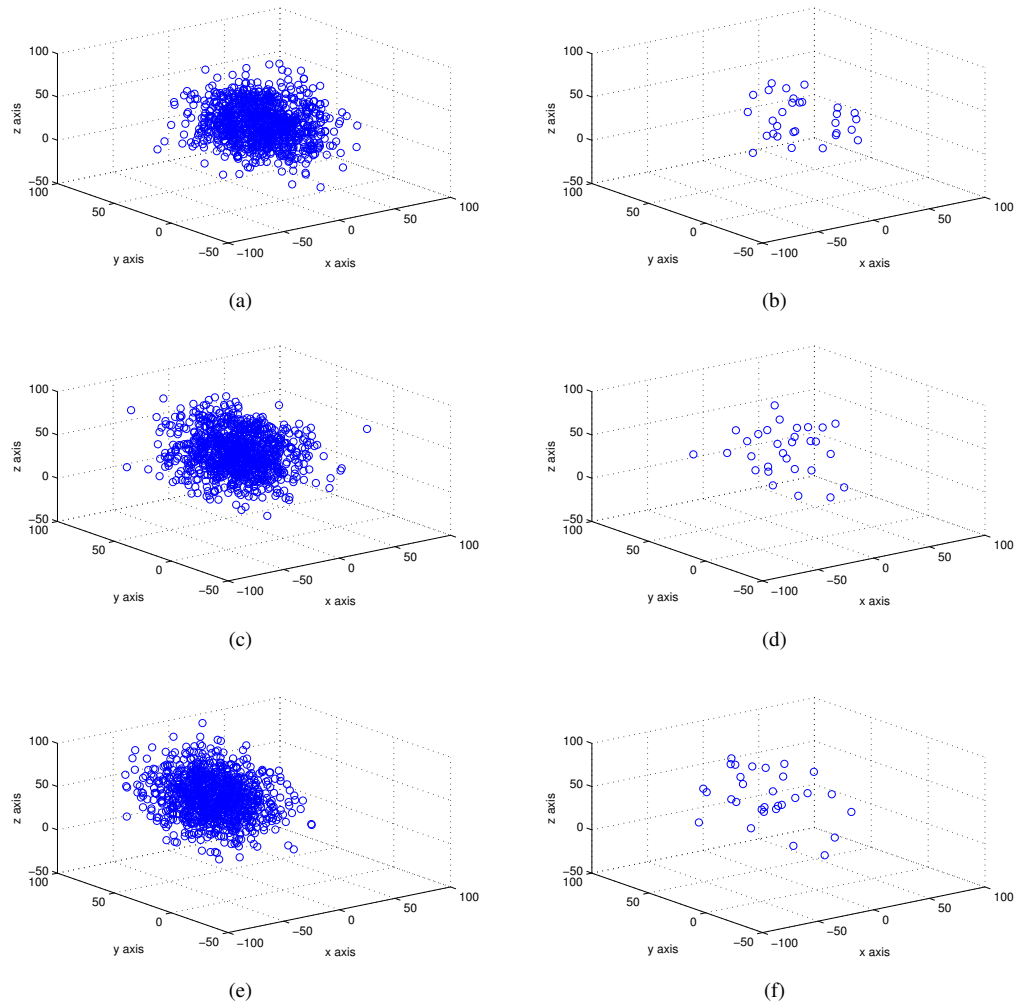


Fig. 8. Showing distribution of agents following a dynamic chemical cloud.

- [4] J.A. Farrell, M. John, L. Xuezhu, L. Wei and C. Ring, *Filament-Based Atmospheric Dispersion Model to Achieve Short Time-Scale Structure of Odor Plumes*, ENVIRONMENTAL FLUID MECHANICS Vol. 2, No 1-2, pp 143-169, DOI: 10.1023/A:1016283702837.
- [5] D. Zazhitsky and D. F. Spears and W. M. Spears, *SWARMS FOR CHEMICAL PLUME TRACING*, In Proc. of the IEEE Swarm Intelligence Symp, 2005.
- [6] Adept, *Autonomous Mobile Robots, P3-DX*, <http://www.mobilerobots.com/researchrobots/researchrobots/pioneer3dx.aspx>, Last Accessed 31 March 2003.
- [7] Sony, *Aido, your artificial intelligent champion*, <http://www.k-team.com/>, Last Accessed 31 March 2003.
- [8] K-Team, *Khepera Robots*, <http://support.sony-europe.com/aibo/index.asp>, Last Accessed 31 March 2003.
- [9] Cyberbotics professional mobile robot simulator, *Webots 6*, <http://www.cyberbotics.com/overview>, Last Accessed 31 March 2003.
- [10] Simbad Project Home, *Simbad 3D*, <http://simbad.sourceforge.net/>, Last Modified Monday 14th September 2009, Last Accessed 31 March 2003.
- [11] Carnegie Mello, Entertainment Technology Centre, *Panda 3D*, <http://www.panda3d.org/>, Last Accessed 31 March 2003.
- [12] Blender, *Blender*, <http://www.blender.org/>, Last Accessed 31 March 2003.
- [13] OpenFOAM, *OpenFOAM: open source CFD toolbox*, <http://www.openfoam.com/>, Last Accessed 31 March 2003.
- [14] J. Oyekan and H. Hu., *Bacteria Controller Implementation on a Physical Platform for Pollution Monitoring*, Proc. of IEEE Int. Conf. on Robotics and Automation, Anchorage, Alaska, USA, pp 3781-3786, May 3-8, 2010.
- [15] R. Craig, *Boids, Flocks, Herds, Schools: A distributed Behavioural Model*, <http://www.red3d.com/cwr/boids/>, Last Accessed 31 March 2003.
- [16] M.R. D'Orsogna, Y.L. Chuang, A.L. Bertozzi and L.S. Chayes, *Self-Propelled Particles with Soft-Core Interactions: Patterns, Stability and Collapse*, Physical Review Letters, The American Physical Society, 2006.
- [15] R. Olfati-Saber, *Flocking for Multi-Agent Dynamic Systems: Algorithms and Theory*, IEEE Transactions on Automatic Control, Vol. 51, No. 3, pp 401-420, March 2006.