Collective fish behaviour

A Hydrodynamic Interaction Model

Andraž Čeh, Gregor Kovač, Jakob Petek in Matic Šutar

Collective behaviour course research seminar report

19. november 2023

Iztok Lebar Bajec | izredni professor | mentor

This initial report presents our examination and analysis of current models of collective fish behavior, focusing on the integration of hydrodynamic interactions. By reviewing existing methodologies and identifying gaps, especially in hydrodynamic factor representation, we aim to enhance the realism of fish schooling simulations. The report outlines the problem, reviews relevant literature, and details our strategy to tackle these challenges. It underscores the progress made towards creating an improved model with potential applications in various fields.

Collective Fish Behavior | Hydrodynamic Interaction Modeling | Computational Simulation |Self-Propelled Particle Models | Behavioral Patterns in Schooling Fish

1. Introduction

Collective behaviour is a branch of computer science that attempts to model, recreate and visualize the behaviours of groups of animals, such as birds, sheep and insects. In this paper we will be focusing on the collective behaviour of fish, called *schooling*. It has been modeled numerous times before, but very rarely do we see that water physics are also taken into account. We will present one way of modelling the effect of hydrodynamics on the schooling of fish. We will also explain all the techniques used and also present what we plan to do in the future to test how this affects fish behaviour and further improve our model. Our research will be based on [1]. We plan to implement everything presented in this paper and further expand it.

2. Methods

In this section we first address previous work in fish behavioural modelling by conducting a literature review. We then explain the theoretical background from our paper of choice.

Literature review. Early studies of swarm behaviour, such as fish behaviour, employed mathematical models to simulate and understand the behaviour. The simplest mathematical models of animal swarms generally represent individual animals as individuals following rules of similar directional movement, proximity and collision avoidance with their neighbours. One such early example is the boids computer program created by Craig Reynolds [2], which simulates swarm behaviour following the above rules.

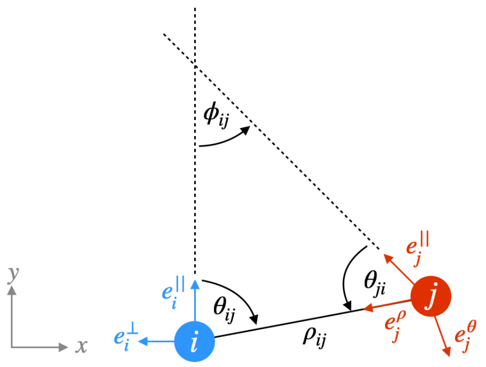
Within the specific domain of fish behavior, some of the early models expanded on the general swarm behaviour and introduced further complexities. For instance, some models focused on calculating velocity and angle based on probability distributions of random influences, as presented in a notable paper [3].

As research progressed, researchers undertook the challenge of modeling specific fish behavioral factors, including schooling, swarming, and milling. A particular study [4] analyzed all conceivable initial states to discern transitions between stationary states, such as schooling, swarming, or milling. A significant finding from this research highlighted that fish density in certain stationary states causes global interactions, where each fish perceives the presence of all others. The swarm algorithm proposed in this study adheres to the Lagrangian approach.

Recently, the research has expanded and includes various machine learning techniques to further improve the understanding and modelling of fish swarm behaviour. A case in point is a recent paper [5], wherein machine learning and computer vision methodologies were employed to track and gather fish pattern data for constructing a fish movement model.

Apart from new emerging modelling techniques, traditional mathematical models often overlook or oversimplify the intricate hydrodynamic interactions among fish. This is where we hope to improve and expand the research by implementing and improving the fish model to support hydrodynamic interactions, as proposed by [1].

Fish behaviour model. To model fish we will use so-called *self-propelled particle (SPP) models*, which can be constructed from simple rules to induce relatively complex behaviour. Each fish will be modeled as a particle moving around in a plane. It will move forward at some constant velocity v. Now we want to introduce interaction between a fish and its neighbours. All of the spacial parameters that we will use to achieve this are shown in the figure 1 and we will use them in the explanation later on.



Slika 1. Visualization of the parameters used to model the interaction between fish i and j.

Firstly, we will add an attraction factor $k_p[m^{-1}s^{-1}]$ that attracts a fish towards its nearest neighbours and an alignment factor $k_v[m^{-1}]$ that makes a fish align with its neighbours. We will also add some Gaussian-distributed rotational noise σ to introduce randomness. This is approximately how fish schools are usually modeled. On top of this, we will add a fish's response to flow disturbance from other fishes. This is represented by an elementary dipole (a flow of a shape that is similar to the shape of a magnet's force field) with intensity Sv where $S = \pi r_0^2$ is the surface of a fish with length r_0. We have previously stated that fish will be modeled as points, but in terms of hydrodynamics we need to model them as objects with a surface. Now we will also introduce some new variables for readability purposes: $I_{||} = k_v\sqrt{\frac{v}{k_p}}$ represents alignment, $I_n = \sigma(vk_p)^{-\frac{1}{4}}$ represents noise and $I_f = S\frac{k_p}{v}$ represents dipole intensity. We can put all of this together to obtain motion equations:

$$\dot{r}_i = e_i^{||} + U_i \tag{1}$$

$$\dot{\theta}_i = \langle \rho_{ij}\sin(\theta_{ij}) + I_{||}\sin(\phi_{ij})\rangle + I_n\eta + \Omega_i \tag{2}$$

The equation 1 represents the movement of a fish from $\dot{r}_i$ at constant speed in the direction of its orientation $e_i^{||}$. We will call U_i the *drift term* that takes into account hydrodynamics. It is defined as:

$$U_i = \sum_{j\neq i} = u_{ji}, \quad u_{ji} = \frac{I_f}{\pi}\frac{e_j^{\theta}\sin(\theta_{ji}) + e_j^{\rho}\sin(\rho_{ji})}{\rho_{ij}^2}.$$

Each fish generates a flow field and u_{ji} is the field velocity generated by the j-th fish, affecting i-th fish. The spacial relation between a pair of fish is represented with polar coordinates in the framework of the j-th fish, hence the angles in the expression.

The equation 2 represents the rotation of a fish. The term η represents a *standard Wiener process* (a stochastic process used to model noise and disturbances) that is multiplied by the noise term. This introduces a model of free will in a fish. Ω_i is the rotation introduced by hydrodynamics and is defined as

$$\Omega_i = \sum_{j\neq i} e_i^{||}\cdot\Delta u_{ji}\cdot e_i^{\perp}.$$

This is essentially the sum of the gradients of velocities of other fish along both parallel and perpendicular directions.

The notation $\langle\star\rangle$ indicates the averaging of all terms over the *Voronoi neighbours* (ν_i) (a selection of neighbours based on Voronoi diagrams) of a fish, weighted with $1 + \cos(\phi_{ij})$:

$$\langle\star\rangle = \frac{\sum\limits_{j\in\nu_i}\star(1+\cos(\theta ij))}{\sum\limits_{j\in\nu_i}(1+\cos(\theta ij))}$$

The model in action. We will be solving the presented system of equations with some numeric methods, since they are quite complex and we will be choosing simulation speed over accuracy. By solving the system for every fish, we will simulate motion that also takes into account hydrodynamics.

But what kind of collective behaviour will we see? [1] states that by setting different values of the initial parameters of $I_{||}$, I_n and I_f we can expect different behaviours of schools of fish:

1. *Swarming* is a behaviour where fish form very sparse groups without a distinct orientation. This occurs when the noise is comparable or greater than the alignment factor ($k_p \leq \sigma$).

2. *Schooling* creates denser groups with a specific traveling direction. This occurs when the alignment factor is bigger than the noise ($k_p > \sigma$).

3. *Milling* is a behaviour where some kind of a vortex is created. This occurs when alignment and attraction are comparable ($k_p \sim k_v$) and noise σ is somewhat low.

4. *Turning* is a behaviour that cannot occur without the use of hydrodynamics. Here a group of fish follows a larger circle trajectory. It happens when the variables $e_i^{||}$, e_i^r and $\dot{r_i}$ reach a specific value.

In our implementation we will try to search for the sets of parameter values where these behaviours occur.

Implementation. For the implementation of the simulation, we had to first decide on the tools we were going to be using. We considered different aspects that could help us favor one set of tools over the other. These aspects were:

- **programming paradigm**

  Because of the nature of the problem, we decided that we wish to develop according to the object-oriented programming paradigm. This will enable us to implement the simulation in a structured way. It is also a programming paradigm we are most experienced with.

- **ease of use**

  One important factor when deciding upon the set of tools is the scope of the project. Because we are planning to create a relatively simple simulation, we value simplicity over scaling and performance.

- **number of features**

  For our implementation, we wish to choose a set of tools that will be a minimal set containing the tools we will actually be using. This will contribute to the light-weightiness of the implementation and possibly to the ease of use aspect (e.g. in game engines you can not always set the time between two steps to be constant. Doing this in e.g. pure C++ is trivial). Since our program will not be very complicated, we expect this set to be relatively small.

- **experience level of our group**

  For the last aspect, we considered the experience of our group with using different sets of tools. Working with tools we are already familiar with will contribute greatly to the reduction of the time needed to finish the implementation.

We considered different *"tools"* such as python, C++, Unity[1] and Unreal Engine 5[2]. In the end, we chose to implement our simulation in python because it was the most suitable for our needs (according to what we discussed above). However, for some components of our simulation it would be difficult and inefficient to implement them with basic python libraries. These components are mainly visualization and implementation of numerical methods needed to solve the equations from section 2.

For the visualization, we have mostly settled on python API for opencv[6]. We have also considered pygame but later concluded that we will not need most of the functionality of that library (it is intended for game development in python).

The choices for the implementation of the numerical methods were SciPy, PyTorch and Tensorflow. Since PyTorch and Tensorflow implement many features connected with deep learning (that we don't need), we decided to use SciPy[7]. If we run into unexpected problems with SciPy (e.g. performance issues), we will later on possibly consider the other two alternatives. These are all the libraries we have already worked with.

[1] https://unity.com/

[2] https://www.unrealengine.com/en-US/unreal-engine-5

3. Results

Our review of existing literature and models of collective fish behaviour has provided a strong foundation for our study. This has deepened our understanding of the complexities of fish schooling, particularly the frequently neglected hydrodynamic interactions.

We have examined Self-Propelled Particle (SPP) theories, combining traditional behavioural factors with new hydrodynamic components.

Key accomplishments at this stage include:

- Examination of existing models and their limitations, particularly in terms of hydrodynamic interactions.
- Development of a theoretical framework that incorporates hydrodynamic factors into fish behaviour modelling.
- Identification of critical parameters (alignment, noise, dipole intensity and hydrodynamic influence) and their roles in shaping collective fish behaviour.

Our current results show promise for a more accurate model of fish schooling, which we plan to further refine and test in the next stages of our research. In our work, we plan to fine-tune our model, perform broader simulations, and align our findings with real-world data to confirm and enhance our method.

4. Discussion

In this section we will first discuss possible limitations of the chosen implementation strategy and possible solutions.

We stated previously that we are going to be using python SciPy library for the implementation of the numerical methods for solving the equations from chapter 2. Depending on the number of particles (fish) we wish to simulate, the usage of a CPU based library for numerical methods may prove itself to be a problem. One way to deal with such potential performance issues would be to use GPU computing through libraries such as CuPy[8]. CuPy enables NumPy and SciPy to use GPU computing. We have not used the library before and it would introduce additional complexity to the implementation, therefore if possible, we would want to avoid using it.

A similar problem could arise when using opencv library for visualization and rendering purposes. The visualization will for the most part consist of drawing sprites on the screen and rendering UI for the real-time control of different simulation parameters. One possible limitation of such an approach could be slower rendering of the sprites, since as far as we know, they will need to be rendered on CPU. It is not yet clear if this will be something we will need to deal with in the future and it won't be until we actually run the simulations. As before, an alternative to using opencv would be using a library for GPU accelerated rendering such as PyOpenGL which is a cross platform python binding to OpenGl[9] and related API. Again, using PyOpenGl would introduce additional complexity to the implementation of visualization therefore we would like to avoid it if possible (it is not even clear if it would improve the performance drastically since, in essence, we only need to make some simple modifications to an image that represents a screen). A simpler approach would be to use the already mentioned library CuPy in combination with NumPy to add the fish sprites to a 2d array representing the screen and then displaying it using opencv.

Another workaround for both of the limitations would be to simply decrease the number of rendered particles. However, there is a limit or a minimal number of fish we need to simulate for the desired behaviour patterns of schools of fish to emerge (we can still render fewer fish than we simulate if the visualization will be a bottleneck in our implementation).

DOPRINOSI. GK wrote the introduction and described the collective behaviour model. MŠ wrote the implementation and discussion. AČ wrote abstract and results. JP wrote the literature review.

Literatura

1. Filella A, Nadal Fmc, Sire C, Kanso E, Eloy C (2018) Model of collective fish behavior with hydrodynamic interactions. *Phys. Rev. Lett.* 120:198101.
2. Wikipedia contributors (2023) Swarm behaviour.
3. Huth A, Wissel C (1992) The simulation of the movement of fish schools. *Journal of Theoretical Biology* 156(3):365–385.
4. Gautrais J, Jost C, Theraulaz G (2008) Key Behavioural Factors in a Self-Organised Fish School Model. *Annales Zoologici Fennici* 45(5):415 – 428.
5. Bhaskaran R, Baskaran RK, Vijayalakshmi C (2020) Machine learning technique for analyzing the behavior of fish in an aquarium in *Modeling, Machine Learning and Astronomy*, eds. Saha S, Nagaraj N, Tripathi S. (Springer Singapore, Singapore), pp. 55–65.
6. Bradski G (2000) The OpenCV Library. *Dr. Dobb's Journal of Software Tools.*
7. Virtanen P et al. (2020) SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* 17:261–272.
8. Okuta R, Unno Y, Nishino D, Hido S, Loomis C (2017) Cupy: A numpy-compatible library for nvidia gpu calculations in *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Thirty-first Annual Conference on Neural Information Processing Systems (NIPS).*
9. Woo M, Neider J, Davis T, Shreiner D (1999) *OpenGL programming guide: the official guide to learning OpenGL, version 1.2.* (Addison-Wesley Longman Publishing Co., Inc.).