FRIteza

Simulating flock of birds using only vision

Žiga Leskovec and Maksimiljan Vojvoda

Collective behaviour course research seminar report

January 7, 2024

Iztok Lebar Bajec | izredni professor | mentor

When modeling collective behaviour it is commonly assumed that agents inherently know other agents position, velocity and direction. There exists a motivation to get rid of these assumptions and model the behaviour based on how internal and external information is acquired and processed. Vision is one of the most important sensory systems that provide crucial external information, which turns out to be sufficient for modeling interactions between agents in a swarm. In this seminar we explore a mathematical framework for perception-based interactions proposed by Renaud Bastien and Pawel Romanczuk.

Simulation | Vision | Flock of birds

Models of collective behaviour often rely on interactions that do not have a direct physical reality (such as neighbour velocity, relative position and direction). One example of this is the simulation of fish schools [1]. However, this assumption of how the information is processed by agents limits our understanding of the underlying complexity that takes place in such phenomena. A better alternative would be to model the behaviour around internal and external information that agents are capable of acquiring.

In this seminar we use a mathematical framework proposed by ~~Renaud~~ Bastien and ~~Pawel~~ Romanczuk [2], to create a simulation of flock of birds that solely relies on vision. We improve on the original paper by replacing ray casting with direct projection, which increases accuracy and computational speed of each simulation step. Finally, we also introduce colors to the visual field projections, which in turn allow us to model more versatile collective behaviours with different agent types (e.g. predator, prey, disruptor ...).

Methods

The simulation is done in 2-dimensional space where agents are represented as simple disk objects with full 360° view. On each simulation step, their velocity is modified based on a projection of their surrounding visual field.

The parameters that encode the severity of the response to the environment can also be upgraded with the use of *colors*, where parameters vary between each color. In other words, a boid can have different responses to different kinds of objects in their surroundings.

This simulates a primitive form of vision.

Visual field projection. Objects around the agent are projected onto their visual field, described by $P(\varphi)$. Function $P(\varphi)$ represents visual obstructions of the agent, where φ is an angle of the visual field. The result is binary, where 0 represents "not obstructed" and 1 represents "obstructed". An example of a visual field projection can be seen on figure 1.

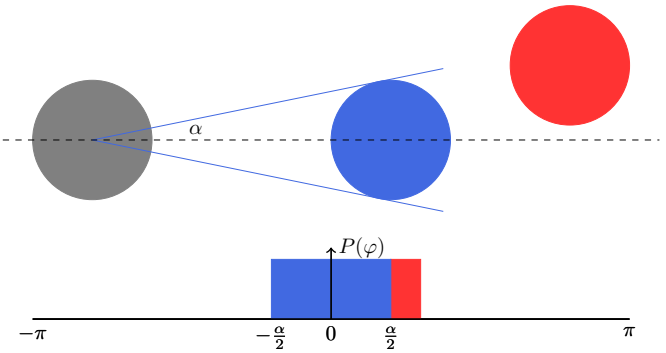


Figure 1. Projection field graph of an object. The projection interval of the blue object overlaps (overrides) part of the interval of the red object.

A model of collective behavior based purely on vision

When modeling collective behaviour it is commonly assumed that agents inherently know other agents position, velocity and direction. There exists a motivation to get rid of these assumptions and model the behaviour based on how internal and external information is acquired and processed. Vision is one of the more important sensory systems that provide crucial external information, which turns out to be sufficient for modeling interactions between agents in a swarm. In this seminar we explore a mathematical framework for perception-based interactions proposed by Renaud Bastien and Pawel Romanczuk.

Simulation | Vision | Flock of birds

Colored visual projection field. After achieving the basic monochrome visual projection field, we can extend it to also hold information about the color of the projected object as shown on figure 1. This will come in handy when implementing different behavioral responses to different surrounding objects. This allows us to implement *disruptor* birds to make the simulation a bit more interesting.

We must be careful to ensure that projection intervals of closer objects take precedent over more distant objects. In other words, we must sort objects based on distance just like in regular computer graphics. Just like in computer graphics, we don't actually sort the objects, we instead keep a z-index (or depth index) of each projection interval and then override intervals with larger z-indexes.

Velocity. On each simulation step, the velocity of agent i, is modified by Δv_i. The general speed delta is summarized with the following equation:

$$\Delta v_i = \mathrm{F}_{\mathrm{ind}}(v_i) + \mathrm{F}_{\mathrm{vis}}(P_i) \quad [1]$$

where $\mathrm{F}_{\mathrm{ind}}$ function represents speed delta collected from individuals "internal information":

$$\mathrm{F}_{\mathrm{ind}} = \gamma(v_{\mathrm{pref}} - v_i)\hat{v_i} \quad [2]$$

where γ represents speed relaxation rate, v_{pref} preferred speed of the individual and $\hat{v_i}$ normalized direction vector. Function $\mathrm{F}_{\mathrm{vis}}$ transforms visual field to the individuals speed delta. It is independent of other individuals properties, and is described with the following equation:

$$\mathrm{F}_{\mathrm{vis}}(P) = \int_{-\pi}^{\pi} G(P,\varphi)\ h(\varphi)\ d\varphi \quad [3]$$

Here $G(P,\varphi)$ encodes how information from the visual field impacts the movement, while $h(\varphi)$ encodes properties of the perception-motor system, in our case, it describes how front-back distance impacts the speed and how left-right distance influences the heading direction of an agent. For convenience the equation is split into two parts

$$\Delta v_i = \gamma(v_{\mathrm{pref}} - v_i) + \int_{-\pi}^{\pi} cos(\varphi)\ \alpha_0 \left(-P_i(\varphi)\ +\ \alpha_1(\partial_\varphi P_i(\varphi))^2\right) d\varphi \quad [4]$$

$$\Delta\Psi_i = \int_{-\pi}^{\pi} sin(\varphi)\ \beta_0 \left(-P_i(\varphi)\ +\ \beta_1(\partial_\varphi P_i(\varphi))^2\right) d\varphi \quad [5]$$

The first (Δv_i) describes speed delta, while the second ($\Delta\Psi_i$) describes heading angle delta. Consequently the heading vector is now removed, since it is encoded as the heading angle.

As we will see further in the seminar, parameters α_1 and β_1 influence the equilibrium distances. α_1 influences the front-back distance equilibrium $\frac{r}{\alpha_1}$, while β_1 influences the left-right distance equilibrium $\frac{r}{\beta_1}$, where r stands for agent radius.

For a more realistic simulation we also added parameter $v_{\mathbf{max}}$ that constraints boid's speed. This was needed when adding *disruptor* boids, as their repulsive/attractive behaviour resulted in unrealistic speeds (compared to v_{pref}).

Integration of P. One of the core elements of the simulation is calculating integrals of functions multiplied with P such as $\int_{-\pi}^{\pi} cos(\varphi)P_i(\varphi)$ from equations 5 and 4. Because the projection function P contains contiguous regions of values 0 or 1, we can split the integral into multiple definite integrals where we calculate only the sections where P has value of 1. A visual representation of multiplying a trigonometric function with P (in this case *cos*) can be seen on figure 2. This trick trivialises the integral calculation.

With colored projection fields, we must also take into consideration that each interval has a different set of constants (α_0, α_1, β_0, ...). This doesn't change much as all we must do is retrieve the appropriate constants of the *colored* interval we are currently handling.

Implementation. The simulation is implemented in C++.

In the source paper [2], the author calculates the projection function P using ray casting. Ray casting is a method of sampling the environment for obstructions by calculating intersections of all objects with a line (a ray). Casting more rays results in a more accurate sampling of the environment. This is also highly inefficient, as one would have to cast an enormous amount of rays to gain a very precise insight into their environment.

We have improved on this by projecting the objects directly by calculating the appropriate angles of vision for each object as show in figure 1. For each object we

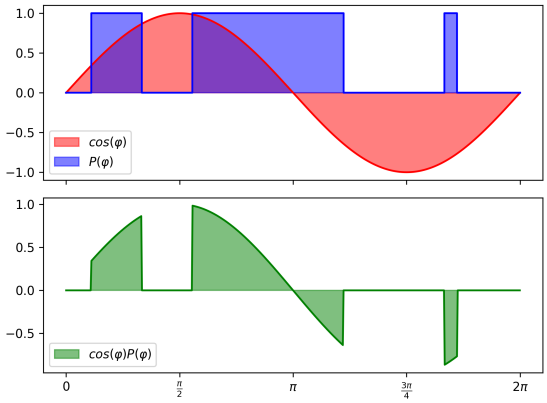


Figure 2. Multiplying a function (in this case cos) with some projection function P.

constructed an array of intervals where P would have a value of 1. By carefully merging these intervals we kept the array *clean* as none of the intervals would intersect. With this array we could trivially calculate all the necessary components described in equations 4 and 5 using the method described in the previous section.

The underlying simulation loop is also parallelizable, as agents are dependant only on the previous state of the simulation, therefore we can use multi-threading to increase the simulation speed, thus allowing us to simulate larger or longer simulations.

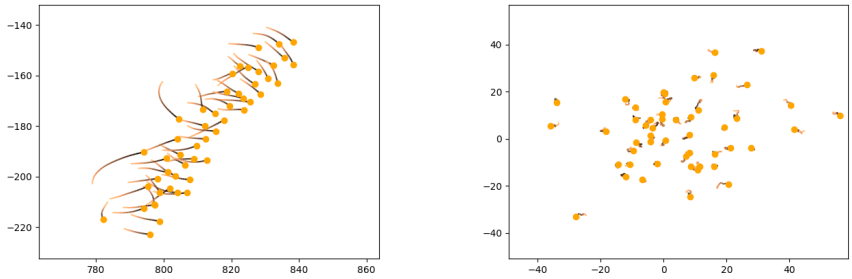
Simulation visualisation. The speed of image generation is not important to this seminar and is therefore done separately in Python with Matplotlib. Every simulation figure has its own one minute long animation available on our Github repository https://github.com/siggsy/collective-vision/tree/main/results. It is worth noting that the size of the agents is not to scale.

Results

Every simulation was run for 2000 steps with 50 agents randomly placed in a box bounded by $(0, 0)$ and $(5, 5)$. All agents had a radius of 0.5 and had v_{pref} set to 0.5. Simulation parameters were set to the following values:

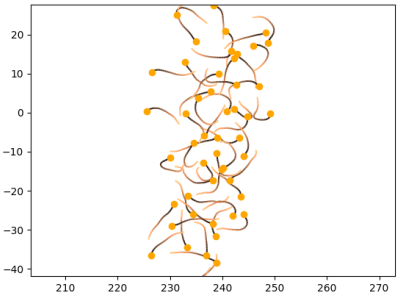
- $\alpha_1 = 0.08$
- $\beta_1 = 0.08$
- $\gamma = 0.95$

In figure 3 we can see 3 different collective behaviours using α_0 and β_0 parameters found in the paper this seminar is based on[2].



(a) Polarized line ($\alpha_0 = 0.5, \beta_0 = 0.01$)

(b) Crystal ($\alpha_0 = 0.1, \beta_0 = 10$)



(c) Swarm ($\alpha_0 = 0.5, \beta_0 = 1$)

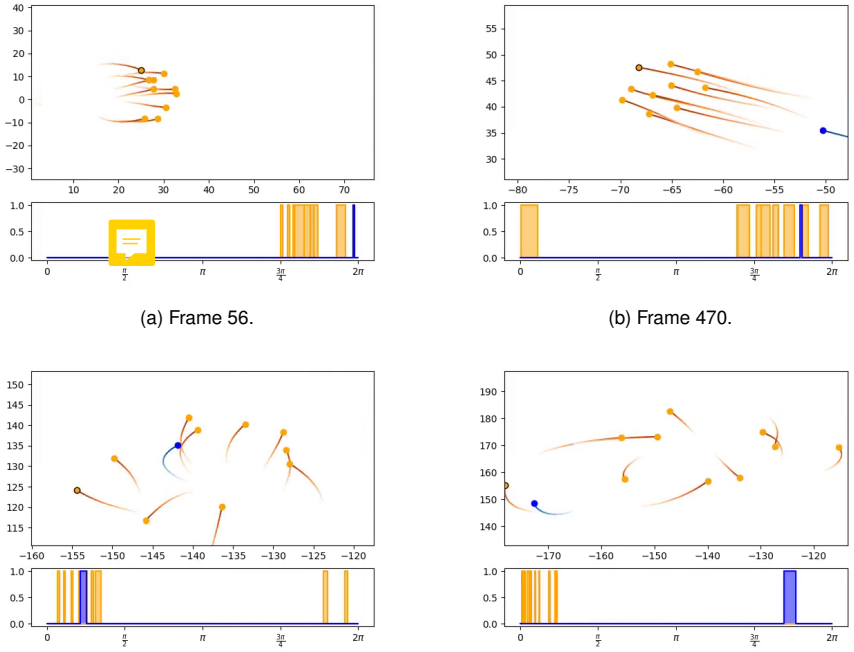
Figure 3. Last frames of the tested simulations

In simulation (a) all agents move in one general direction and form a line. In contrast, in simulation (b) agents shake in-place and barely make any distance. Since the figures represent the last frames of the animation, this can be observed by checking their final coordinates. In the final simulation (c) agents are moving around freely while maintaining a swarm.

When observing these behaviours, we can intuitively understand parameter α_0 as boid's drive to maintain equilibrium distance by changing its speed, and parameter β_0 as boid's drive to maintain equilibrium distance by changing its heading direction.

Next we tried a simulation with one *disruptor boid (**blue**)*. The parameters for orange boids were similar to the ones for a polarized line ($\alpha_0 = 0.5, \beta_0 = 0.08$). The new blue boid has a strong attraction to the orange boids and has higher maximum velocity. On the other hand, the orange boids strongly avoid the blue boid.

A few frames of the simulation can be seen on figure 4, where each frame also shows a graph of the projection field of the outlined orange boid. In frame (a), the orange boids have not yet encountered the blue boid and continue towards the right. In frame (b) they finally meet and turn around to try to fly away. In frame (c) the blue boid finally catches up to the other boids and disperses them. Frame (d) shows the general chaos of the whole flock as the blue boid franticly tries to catch an orange boid.



(a) Frame 56.

(b) Frame 470.

(c) Frame 695.

(d) Frame 912.

Figure 4. A simulation with an added graph of a projection field of one boid.

Discussion

In this seminar we successfully implemented and reproduced different collective behaviours of birds using only their vision as described in the original work by Bastien and Romanczuk [2]. The implementation was improved by replacing ray casting with direct projection. This increased the accuracy and calculation efficiency of the visual fields of the birds.

We have also implemented the *color* information of each projection interval as that allowed us to implement different kinds of boids (regular boids and disruptor/predator boids), as originally proposed in the paper but not implemented.

The simulation is limited to only 2 dimensions as visualisation of more would be quite cumbersome.

The whole project is available on our GitHub repository https://github.com/siggsy/collective-vision.

CONTRIBUTIONS. Žiga Leskovec fine tuned the simulation parameters and handled the simulation visualisation; Maksimiljan Vojvoda implemented the simulation loop.

Bibliography

1. Huth A, Wissel C (1992) The simulation of the movement of fish schools. *Journal of Theoretical Biology* 156(3):365–385.

2. Bastien R, Romanczuk P (2020) A model of collective behavior based purely on vision. *Science Advances* 6(6):eaay0792.