COP290 PROJECT

Starling Murmuration

Shubham Jain
Akshay Neema

Supervisor: **Prof. Subhashis Banerjee**

Department of Computer Science and Engineering
IIT Delhi

IIT Delhi May 2018

Table of contents

Abstract

We aim to study the collective behavior of huge swarms of birds (Starling Birds). Their flocking maneuvers are of dazzling complexity in their changes in density and flock shape, but the processes underlying them are still a mystery.Murmurations exhibit strong spatial coherence and show extremely synchronized manoeuvres, which seem to occur spontaneously, or in response to an approaching threat, like hawks or peregrine falcons.

We aim to model whether such complex patterns can emerge by self-organization. In our computer model, we combine the usual rules of co-ordination based on separation, attraction, and alignment with specifics of starling behavior:
Our model generates patterns that resemble remarkably the qualitative flight pattern of the Starling Birds.

Chapter 1

Understanding Starling Murmuration

1.1 The reasons behind Group Formation

There are a number of different reasons, most of which centre upon managing predation risk. These are:-

1. The larger group you are in, the better the chance someone else will get eaten if a predator attacks. This idea, known as the selfish herd, is a favourite explanation in undergraduate evolutionary biology courses for grouping behaviour.
   Starlings are preyed upon by hawks and falcons, and it is plausible to consider a murmuration as a continuous movement towards the safety of the centre. As a result, the centre never stabilises and the murmuration twists and turns in a perpetual escape motion.

2. Also, it could be that the murmuration itself provides a way of monitoring predators as they approach. Work in the 1970s[Ref.1] showed that starlings in larger groups responded to the presence of a model hawk faster, and recent work has shown that the formation of 'waves' in murmurations is linked to reduced predation success by peregrine falcons. Waves propagate away from an attack, and so fluctuations in the local structure are likely also to be efficient in confusing potential predators.

1.2 What are the rules of Interaction?

There are many questions about the in-flight interactions of the starlings. How many neighbours do starlings interact with? Do they try to take the same heading as their neighbours or are they simply attracted to them?
Starlings seems interact with multiple neighbours, using a variety of sense data. In our model, we tend to accelerate them towards the center of mass in a particular area around the starling. Regarding the study of flocking behaviour, one of the first to start exploring the potential and possibilities of this topic was Craig W. Reynolds who, in 1987, published his paper on "Flocks, Herds, and Schools; A distributed Behavioural Model". In this paper, Reynolds defined some fundamental aspects . Reynolds defined three empirical rules:

1. Flock centering: The desire for agents to stay together with and nearby other agents.
2. Collision avoidance: Avoid collision when agents are close to each other.
3. Velocity matching: Attempt to keep similar velocity and direction as other agents.

The behaviours that make up Reynolds flocking model are based on the principle that each agent relates to its neighbouring agents. We use this information in our mathematical model.

Chapter 2

Model of Starling Murmuration

2.1 Rules of Interaction

Numerical models of self-organized motion, inspired both by biology and physics support the idea that simple rules of interaction among the individuals are sufficient to produce collective behavior.
The main theoretical assumptions (attraction among the individuals, short range repulsion, and alignment of the velocities) are quite reasonable.

1. **Cohesion:-**The main goal of the interaction among individuals is to maintain cohesion of the group. This cohesion is a very strong biological requirement, shaped by the evolutionary pressure for survival.

   To grant cohesion, we make the assumption that individuals attract each other. The distance model we take is the same as in physics, i.e., metric distance and the cohesion force only considers birds within their neighborhood. The fact that birds on the outside of the flock have a stronger desire to cohere than birds on the inside, who are already close is taken into account.

2. **Separation**:- The separation is force is best understood by its effects within two different areas. If the neighbor is closer than r_{sep} = seprationRadius, then there is a strong force encouraging it to separate from the neighbor and that such interaction decays with increasing distance between individuals.This contributes significantly to the separation and avoids collisions.Then, past r_{sep}, there is no separation force.

3. **Alignment:-** The alignment neighborhood is almost identical to the cohesion neighborhood, but without the inner sphere where the force is zero. It is also the simplest force to describe and steers the bird towards the average forward direction of its interaction neighbors. It is calculated by giving it a force in the direction of the difference between its direction and its interaction neighbors. This is calculated by looking at the normalized forward vectors for the birds.

2.2 Algorithms for the rules of Interaction

Some of the main parameters that are needed by each of the rules to govern the movements of boids are:-

1. **Location:-** The x , y and z coordinates of the current position of the boid(Starling Bird).
2. **Velocity:-** The velocity vector of the boids.
3. **Friends:-** All those boids which are in interacting range of the boid of concern.(This drastically reduces the time complexity.)

Also, to make simulation more realistic , we have put an upper limit on how fast the boids can move and turn . For each boid we perform all the 3 rules of interaction as discussed already and then update their velocity and position vectors accordingly.

We now present the 3 main algorithms for the interactions among the boids.

Algorithm 1 Movement of Boids

```
1:  procedure MOVEMENT(Boids)
2:
3:      BoidVector
4:      Vector pos_b
5:      Vector vel_b
6:      Timer
7:
8:  for each b in BoidVector:
9:
10:     friend_b = b.friends()
11:                    ▷ (Actually, in our implementation friend_b is global variable itself.)
12:
13:     vel_b += b.cohesions()
14:     vel_b += b.align()
15:     vel_b += b.sepration()
16:
17:     pos_b = pos_b + delta(Timer) * vel_b
18:     vel_b.update()
19:     pos_b.update()
20:
21: end for:
```

Algorithm 2 Cohesion Algorithm

```
1:  procedure COHESION(boid b_j)
2:      Vector v_j                                          ▷ Initialize (0,0,0)
3:      int count                                           ▷ Initialize 0
4:      float dist                                          ▷ Initialize 0.0
5:      for each boid in friend_b do                        ▷ Here we do the velocity update
6:          dist = ( mod ( diff ( b.pos, boid.pos ) )
7:          if ( dist < cohesR and dist > cohesR / 4 ) then v_j += boid.pos and count++;
8:
9:      if count > 0 then v_j = v_j / count
10:     v_j =v_j / cohesionRadius
        return v_j
```

Algorithm 3 Sepration Algorithm

```
1:  procedure SEPRATION(boid b_j)
2:      Vector v_j                                          ▷ Initialize (0,0,0)
3:      int count                                           ▷ Initialize 0
4:      float dist                                          ▷ Initialize 0.0
5:      for each boid in friend_b do                        ▷ Here we do the velocity update
6:          dist_v =( diff ( b.pos, boid.pos ))
7:          dist = magnitude(dist_v)
8:          if ( dist < seprR ) then v_j += normalize(dist_v)
9:
10:     if count > 0 then v_j = v_j / count
        return v_j
```

Algorithm 4 Alignment Algorithm

```
1:  procedure ALIGNMENT(boid b_j)
2:      Vector v_j                                          ▷ Initialize (0,0,0)
3:      int count                                           ▷ Initialize 0
4:      float dist                                          ▷ Initialize 0.0
5:      for each boid in friend_b do                        ▷ Here we do the velocity update
6:          dist = ( mod ( diff ( b.pos, boid.pos ) )
7:          if ( dist < alignR) then v_j += boid.vel and count++;
8:
9:      if count > 0 then v_j = v_j / count
        return v_j
```

References

[1] Andrew J. King and David J.T. Sumpter,
http://www.collective-behavior.com/publ/KingSumpter2012.pdf

[2] Charlotte Hemelrijk
https://www.technologyreview.com/s/415022/first-simulation-of-the-flocking-behavior-of-starlings/

[3] H. Hildenbrandt, C. Carere, C-K. Hemelrijk *https://arxiv.org/abs/0908.2677*

[4] https://en.wikipedia.org/wiki/Flocking_(behavior)

[5] *https://en.wikipedia.org/wiki/Starling*