Herd Research

Olivia Wilburn

January 28, 2021

:

Abstract

Numerical modeling of herding and flocking behavior of animals can lead to greater understanding of these patterns, which can have important impacts on a number of other fields such as the physics of self-assembling systems, vehicular traffic, and robotics. The simulating of modeling and understanding flock behavior is an essential aspect of understanding the complexity of life, and can model other phenomena. The simulation code created for this project calculates individual behavior based on a set of psychological forces that exhibits the emergent behavior of group-herding patterns. An added force of a predator is explored that can be created with distance and direction of acceleration. The analysis shown here explores previous studies of flocking behavior and presents insights into a personal study as well as further uses and other studies and their contributions to this topic.

1 Introduction

To create an accurate representing of herd animals and how they operate we needed an understanding of the way they act and those causes. We decided to categorize each decision/mentality as a force in order to represent these decisions precisely using math. Using Boid's Algorithm, and consulting scholarly articles on herd animal movements, we created four main forces. The four forces we decided would represent the animals behaviour the best were center seeking, velocity matching, collision avoidance, and a maximum velocity limit. An additional force representing the avoidance of predators will be explored conceptually at the end of this paper.

This type of simulation is helpful for the modeling of herd behavior, but for other types of analysis as well. The simulating of multiple individual data points can be useful for much broader contexts, such as traffic, clouds, and other groups of particle-like points. Being able to simulate herds specifically is important for the use of herd research as well as the possibility of adapting the code to fit other uses. Boid's algorithm, created in *Flocks, Herds, and Schools: A Distributed Behavioral Model*[1] specifically, helped a lot in the creation of this research as a point of inspiration and direction. Although it's not possible to add the full animation in this report, the image sequences are displayed giving an accurate representation of the changes in data.

The fundamentals behind clustering and, specifically, Boid's Algorithm has influenced a plethora of other studies and scholarly works. In recent years, the amount of internet data and information has skyrocketed. To combat this overload of information, web recommendation has become more popular, which recommends certain pages formed from previous searches and browsing history [2]. Clustering helps divide and sort information about browsing behavior, and Boid's Algorithm helps separate this. The clustering is very mixed, as the each individual data point is considered a Boid. To group these Boid's, two parameters are used: affinity and centroid based calculation. The affinity calculation concentrates on similarities between two boid's at a time, and is higher the more amount of similar objects. The centroid and merging rule defines every boid as a centroid, and then applies a merging process. To merge multiple boid's into one centroid they must be near each other and within the the sight area of one another. Then, the probability, Pmxy, of the two groups x and y become one centroid [2].This probability is also proportional to the affinity of the two boid's. In the case that a boid doesn't belong well to a group, they may leave to look for a group with more similarities. These two parameters and their formulas give the object represented, the centroid, and the centroid with the greatest affinity. The three rules these centroid obey are parallel to the idea of Boid's Algorithm—separation, alignment, and cohesion. This all comes together to sort web recommendations, helping group the huge amount of information present on the internet today. Although, web browsing does not initially seem similar to flocking birds, Boid's Algorithm is very important in this field to continue to tackle the surplus of data.

Robots and automated machines are another form of technology that rely on algorithm's to run. Kasper Støy from the University of Southern Dakota published a paper describing communication forms in multi-robot systems, and the importance of choosing a form that doesn't separate the meaning of messages from the physical environment [3] In discussing a way to simplify path planning, the use of Boid's Algorithm was plausible. The task of making a robot team stay together is similar to that of a herd, with the need to not bump into one another and stay uniformed. Boid's Algorithm provides a simple way to mimic the behavior needed—which in simulation is produced well. However, through this study it was shown that this algorithm doesn't transfer to robots easily. The sensors on the robots are hard to map, for the speed and distance of others, as well as the precision of these variables [3]. Although Boid's Algorithm doesn't work well for these multi-robot systems, it produces the possibility of the involvement of Boid's Algorithm in future robotics and other ways to solve these complex problems.

While Boid's Algorithm has been used in numerous ways as itself, as seen in the robotics field and web information, different variables and situations have been added to the original algorithm to make it more specific. In one instance, a herd of animals with artificial fear was created using Boid's Algorithm. Deer were created as a group of flocking animals, then and autonomous agent was incorporated to simulate fear. This study enhanced the world of virtual environments by including the reaction of behaviors [4]. Another project directed Boid's Algorithm in a different direction. In the study *Autonomous Boids*[5] an additional force is added defined as the change of leadership. This creates the chance for a bird to steer away from the group and escape. Boids on the outside of the group have a higher chance of escaping, although it's not a guarantee [5]. These types of changes to Boid's Algorithm show a few of the many possibilities that research can be expanded from this code.

In the following section the Selfish-Herd Theory is explained, and it's role in natural her behavior. Boid's Algorithm is then discussed in-depth, and compared to the herd research. We then explain the methods used to create the code for the herd research completed and a description of the forces. The results of the code are then discussed with images to further explain. The full code is shown present in the appendix, showing the forces used used to simulate herd behavior.

2 Method

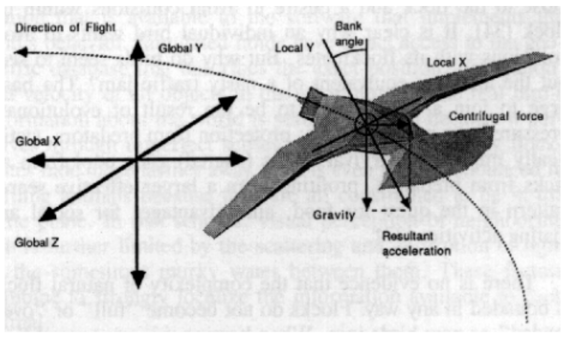
2.1 Selfish Herd Theory

The center seeking force, Force 1, was the first to be calculated for the data. The force pulls the data into the center of the group, by calculating the sum of all the data points location and averaging it to find the mean and average center of the herd. This is illustrated by the 'flowing' movement of the data as each point has their own velocity that is continuously being calculated to find the new center. As the group of data (herd) moves around each animal is at all times trying to be in the middle of the pack, causing the herd to mimick a pulse from a distance. This demonstrates the want for the animal to be in the center of the pack/herd, which is called Selfish Herd Theory. Selfish herd theory was developed to explain this want that animals posses to be in the middle that seems to be hardwired into their brain and that "...there are spatial benefits to individuals in a large group, since individuals can alter their spatial position relative to their group-mates and any potential predator, thus reducing their predation risk." [6]

2.2 Boid's Algorithm

In Reynold's book "Flocks, Herds, and Schools: A Distributed Behavioral Model 1" published in 1987, where he explains his approach to simulate flocking. "One area of interest within computer animation is the description and control of all types of motion...It is not impossible to script flock motion, but a better approach is needed..." [1] Craig explains. This was his inspiration for the code, as motion was a very difficult task to perfect. Computer animators were having a very hard time trying to simulate the complex world of flock of birds. Each bird makes it's own decision, and no two motions birds are exactly the same. However, when in a flock, they're very synchronized. To animate the decisions for each bird while simultaneously having them aligned is not an easy feat. Reynolds created a way to simulate bird flocking as "...the result of the interaction between the behaviors of individual birds."[1] by simulating the behavior of each bird and from their perception.

Reynolds wasn't the first to create a simulation of flocks of birds, the Electronic Theater at SIGGRAPH '85 created a code informally known as the 'Force Field Animation System' which relied on a matrix operator that transformed from a point in space to an acceleration vector, that the birds would then follow linearly. Karl Sims from MIT created a different animation of moving objects, but neither group produced something as "...organized as flocks."[1]. The Boid Flock Model that Reynolds created, he describes as a slight generalization of particle systems. These particles systems have modeled other systems that similarly have individual particles that make up systems that behave alike–clouds, ocean spray, fire, etc. Although the Boid Algorithm is slightly different, it's concept is very similar and those other projects offered inspiration. To simulate the flock, the three behaviors were created: Collision Avoidance, Velocity Matching, and Flock Centering. Extra research on geometric flight was also necessary to simulate the use of an X,Y, and Z, axis, as a bird flying is able to bank turns and has centrifugal force depending on the angle of the bank. The figure shown below describes part of the force considered when Reynolds researched geometric flight.[1]



Forces of Geometric Flight [1]

This flocking model doesn't represent real senses, such as vision and hearing, from animals (or incorporate them into the simulation), but rather approximates the behavior that would occur. Each of the three behaviors acts as an acceleration request of a three-dimensional vector, which each has it's own parameters. When the code is run, all of the different accelerations and runs through them, while sometimes potentially conflicting, which then produces the final decision of the bird. Through most scenarios this worked, the accelerations would sometimes 'cancel out' and the product would result in a slight change of direction. However, in specific scenarios, like avoiding a collision with an object, the averaging of acceleration's wouldn't account for this necessary change, resulting in a collision. To combat this, prioritized acceleration was developed. This orders the behaviors in a list, accommodating for an emergency situation. In this situation, instead of all accelerations averaging out, the most important would take control, leaving the others less 'strength' to override it. These edits helped create one of the first successful codes for flocking behavior that was able to maneuver objects successfully and fluidly, truly representing a group of particle-like points that moved synchronously enough to change direction as one.

2.3 Personal Research

For our personal research in this field, we decided to undertake creating a herd theory. The goal was to create a number of forces that all herd animals experience, affecting their movement and choices when moving in a herd. Some of these forces overlapped with the concept of Boid's algorithm, but not all of them, as birds behave slightly differently than herd animals on land.

Force 1 was described earlier, as the center seeking force. This was created in mind of the Selfish Herd Theory. The Selfish Herd Theory is a consistent action performed by herd animals, centering themselves in a herd. Herd animals feel safer in the middle of the herd, and are always trying to get to that middle spot. To represent this in the code, the force calculated the average position for all the data and found the average center through this. Then as the data would move it would not only have it's regular velocity, but to seek the average location for the group.

Force 2 accounted for velocity matching. While every data point had it's own spatial location and velocity, as a herd moving together, the velocities needed to be matched. This is a common force for any animal whether it be flocking, herding, or hunting. In order to be affective as a group, the group needs to stay close with similar parameters.

Force 3 represents collision avoidance in this program. Collision avoidance has been studied to be the ability, and want, to avoid each other physically. The collision avoidance force numerically changes depending on the animal. For example, cows knowingly bump into each other a lot–never completely through one another–but are able to slightly collide, while other animals are less likely to do this, and have a higher radius of personal space.[7] This force shows the ability to synchronously move together, without full overlap of an exact spatial location.

Force 4 sets a maximum velocity limit. Compared to the previous forces it doesn't affect the look of the herd when moving, but rather prevents chaos from the data going too fast. Biologically speaking, there is a maximum velocity any animal can reach. By setting the limit, it limited the data from being infinitely fast which we know to be impossible in the animal kingdom.

To represent these forces, we chose Python because of it's ability to function as a high-level programing language that supports modules and packages (that were both needed to create this simulation). Using modules and packages allowed for extra functions to be added to create the best working program. For this research extra packages were needed for the math and graphics both involved.

To create the borders of the image sequence we also needed to set parameters to 'corral' the data points in. These were set to reflect the data points in order to create an ongoing loop of movement. Without the reflect order once the data moved out of the borders set the simulation would end.

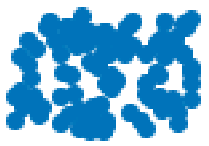
There were several complications running the code once it was developed. Python is a high-level programming language and was able to create the code with the right imports, however it's not made to run advanced animation. To combat this problem multiple approaches were taken. Multiple packages were downloaded to try and allow ImageSequenceClip to compile produced images together. When running the program, it would run itself for a set amount of times, each time producing a new image slightly altered from before. When these images were compiled in a fast sequence, it produced a short animation, similar to a flip-book style. Unfortunately, this ImageSequenceClip function didn't work on all forms of MAC OS and was unable to run multiple times. To combat this we tried a number of other programs meant to compile images. Eventually, we found an a program online that was able to take the images produced and run them to create the visual's shown in this report.

3 Results

The images below were taken at intervals of 3 seconds apart, with the average image being 10 images apart from the previous. This shows the change in animation over time, as the group imitates pulse-like behavior while simultaneously moving as a group. Although it's difficult to illustrate the motion across the axis', it is clear to see the movement inside the herd.

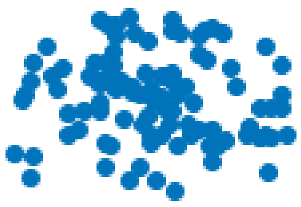
3.1 Herd Simulation Images

3.2



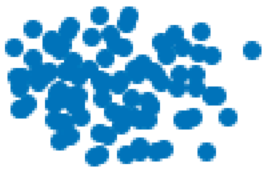
Time 0:00 (s)

3.3



Time 0:09 (s)

3.4



Time 0:21 (s)

4 Conclusion

The final code produced shows a clustering behavior that is similar to observed herd behavior. Center seeking, velocity matching, collision avoidance, and a maximum velocity parameter were all needed to represent their behaviors, and did model these specific forces. With the understanding and analysis of Boid's Algorithm, we were able to use that knowledge to help build the new code and apply certain ideas from it. Although Boid's Algorithm is applied to flocking behavior, there exists few differences. The addition of three-dimensional vectors and an x,y,z for spatial locations when addressing birds being one of them, as they have the ability to cross into all three axis. Herding animals do not have this ability, however, they share similar behavioral traits in two-dimensions. The current code has a very similar model to the Boid's Algorithm model. With further work and an addition of a fifth force, a predator, this could tackle the idea that Boid's Algorithm doesn't represent vision. With a 'knowledge' of the predator force and a reactionary action, this would elevate the research into representing more behavioral components and further simulating herd animals. Similarly, the same new force could be applied to flocking in a three-dimensional way to extend the programs representation. With more research done, this code can continue to be manipulated and improved.

When compiled, the images produced

from this code do a good job of simulating herd behavior with all four components combined. It is difficult to differentiate between each data point in the still images, but when compiled it is easy to see Force 1 taking effect, as the dots take turns pushing to the middle. Force 2, velocity matching is shown as the dots continue group, as none of them are outliers. Velocity matching makes sure the dots are going the same speed and direction—because they have different starting positions they vary slightly in spatial location—but remain as one group when they move across the axis. Collision avoidance, Force 3, is partially acknowledged in these images. Some of the data points overlap, which would represent cows or other animals bumping into each other as their flight distance is minimal. This force would need to be altered if the animal represented was more skittish (for example, a deer) so the overlap wouldn't happen at all. Although overlap can be seen, there is no one-point that is in the exact same location as another—modeling the boundaries of an animal. Force 4, the limit of acceleration can't be demonstrated through the still images, but is represented in the compilation of the images, as they slowly move.

To improve this code and it's ability to simulate herds, in the future we hope to add a fifth force that represents a predator (Force 5). This Force 5 contains three main components. Adding this Force 5 would also reinforce the Selfish Herd Theory. Animals naturally try and center themselves in the middle of the herd, as it's the farthest away from any danger. When a predator does put the herd in danger, this theory is just as strong, as the threat of danger is higher, every animal wants the highest chance of survival. Since this stays the same, there is no extra code needed, but an understanding of why the herd continues to follow Force 1.

The first objective, would be the increase in flight zone distance, an animals 'personal space'. The more scared the animal is, the more the flight distance increases. If the predator is a bigger threat, the distance also increases. To illustrate this through Python, the collision avoidance, Force 2, needs to change, depending on the choice of animal and predator. When they're scared, this distance continues to increasing varying on the degree of anxiety. To represent a bigger threat, with a separate data point, the collision avoidance needs to increase significantly when this point closed in on the herd. If the predator was less of a threat Force 2 would still increase, but slightly less.

Secondly, to more accurately incorporate this predator we need to change the direction of the animals depending on the direction of the predator. This illustrates a reaction between noticing a predator and going the other way. Herds react in two types of ways when confronted by a predator. If the predator comes from behind them or the side, they tend to go directly opposite, linearly, away. However, if the predator faces the herd head-on, it is normal for the herd to split, running away from where the predator is at a 45 degree angle. This angle is created from the forward velocity the animals already have, and their instinct to run directly to the side when confronted, adding the two vectors results in the 45 degree angle that is seen so much in the wild.

Lastly, we would need to increase the acceleration function to increase dependent on distance from the predator. The function would still need a maximum limit, to ensure the animals can't run infinitely fast, but the predator data point gets closer the acceleration would increase and vice versa. With these three components, an accurate representation of a predator would be added to the simulation of actions of natural herd, bettering this code for all future needs.

Outside of bettering the code to improve it, modeling individual data points in groups like this research has lots of possible future applications as well. Certain scientists work with herd animals, and having a program that can simulate these animals and their behavior can be greatly beneficial. Outside of the world of herding animals, the basics of this code can be greatly applicable to a lot of situations. Modeling individual data points that act as a group can be rewritten with different 'behaviors' to represent a plethora of other situations. Traffic is another situation where each data point makes it's own decision, but as a whole every vehicle tends to follow a certain set of rules. Although this research is specifically for herding animals, herd behavior is common throughout the world in every day situations. Being able to model these accurately can not only help further research but further the understanding of the way the world works. Accuracy presents another question for further research–to pursue the unit of measurement research can be compared to natural phenomena.

5 Acknowledgements

I'd like to thank Dr. Eric Edlund for collaborating with me on this project. Without him this research would not have been possible. Also, without the previous research creating Boid's Algorithm this would've been a much more difficult project.

6 References

[1] Reynolds, C. W. "Flocks, Herds, and Schools: A Distributed Behavioral Model." Computer Graphics, 21(4), July 1987, pp. 25-34.

[2] Suguna, R., Sharmila, D., "An Efficient Web Recommendation System using Collaborative Filtering and Pattern Discovery Algorithms," International Journal of Computer Applications (0975 – 8887), Volume 70– No.3, May 2013

[3] Støy, K., "Using Situated Communication in Distributed Autonomous Mobile Robotics," The Maersk Mc-Kinney Moller Institute for Production Technology, University of Southern Denmark, Odense, Campusvej 55 DK-5230 Odense M Denmark

[4] Delgado M. C., Ibanez J., Bee S., et al. (2007), "On the use of Virtual Animals with Artificial Fear in Virtual Environments", New Generation Computing 25 (2): 145– 169.

[5] Hartman C., Benes B. (2006), "Autonomous boids", Computer Animation and Virtual Worlds 17 (3-4): 199–206.

[6] Hamilton, W.D., "Geometry for the selfish herd," J. Theor. Biol., 31 (1971), pp. 295-311

[7] Grandin, T.,"Understanding Flight Zone and Point of Balance for Low Stress Handling of Cattle, Sheep, and Pigs,"Dept. of Animal Science Colorado State University, May 2019

[8] Anderson, R.M., "Formation Flight as a Cooperative Game," Virginia Polytechnic Institute and State University, August 1998

[9] Cui,X., Gao,J., Potok, E.T "A flocking based algorithm for document clustering analysis," Oak Ridge National Laboratory, April 2006

[10] Darling, F.F.,"A herd of red deer: A Study of Animal Behavior," Luath Press, 2008

[11] King, J.A,Wilson, M.A., Wilshin D.S., Lowe, J., Haddadi, H., Hailes, A.S.,

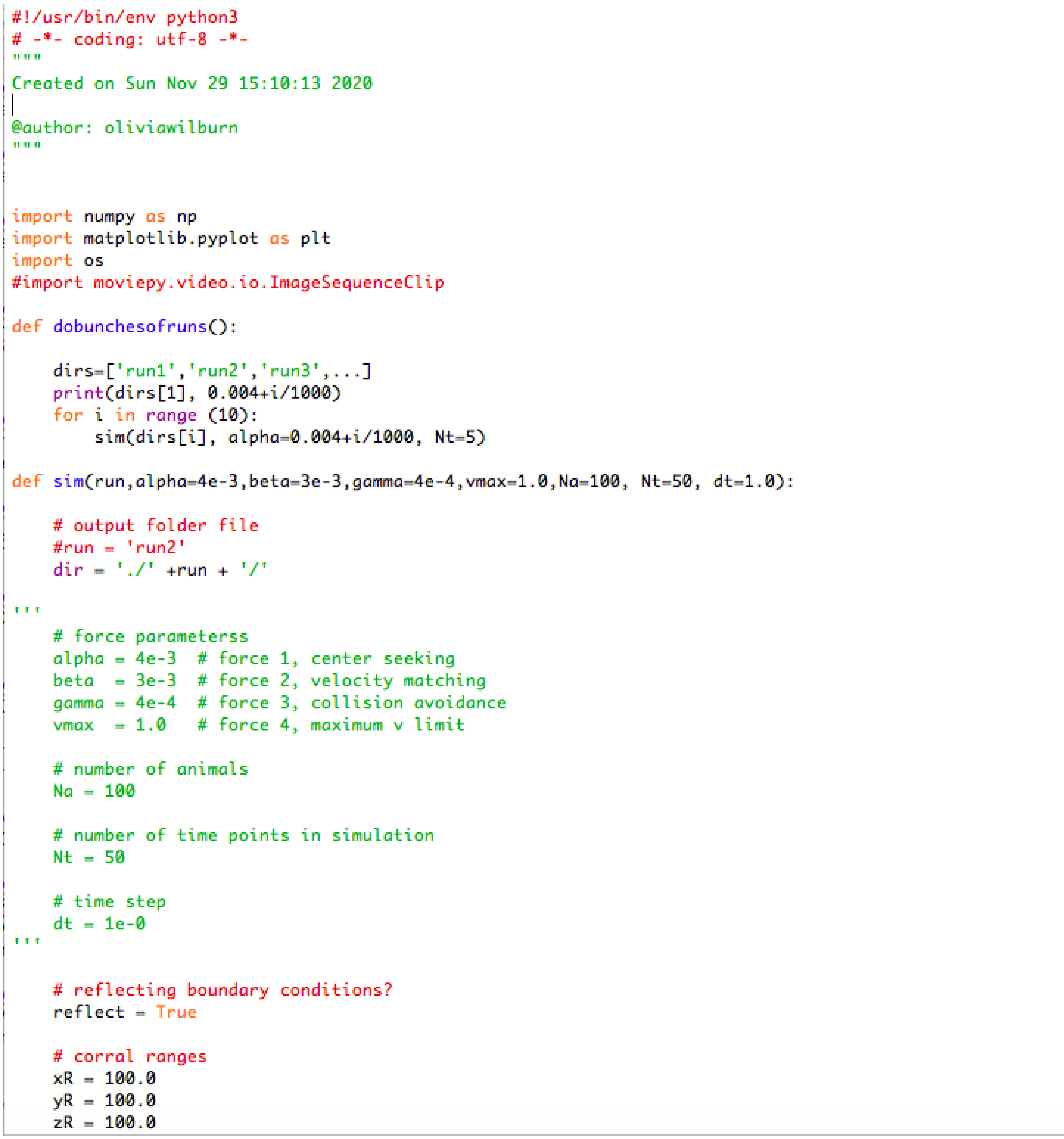
Morton,J., "Selfish-herd behavior of sheep under threat", Current Biology, Volume 22, Issue 14, 2012, Pages R561-R562

[12] Overleaf, "Latex Basics," Overleaf 2020

[13] Ruiz-Vanoye, A.J., Díaz-Parra, O., et al.,"Meta-Heuristics Algorithms based on the Grouping of Animals by Social Behavior for the Traveling Salesman Problem,"International Journal of Combinatorial Optimization Problems and Informatics, Vol. 3, No. 3, Sep-Dec 2012, pp. 104-123

[14] Vaughan, R., Sumpter, N., Henderson, J., Frost, A., Cameron, S., "Experiments in automatic flock control," J. Robot. Auton. Syst., 31 (2000), pp. 109-117

7 Appendix A



```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sun Nov 29 15:10:13 2020

@author: oliviawilburn
"""


import numpy as np
import matplotlib.pyplot as plt
import os
#import moviepy.video.io.ImageSequenceClip

def dobunchesofruns():

    dirs=['run1','run2','run3',...]
    print(dirs[1], 0.004+i/1000)
    for i in range (10):
        sim(dirs[i], alpha=0.004+i/1000, Nt=5)

def sim(run,alpha=4e-3,beta=3e-3,gamma=4e-4,vmax=1.0,Na=100, Nt=50, dt=1.0):

    # output folder file
    #run = 'run2'
    dir = './' +run + '/'

'''
    # force parameterss
    alpha = 4e-3   # force 1, center seeking
    beta  = 3e-3   # force 2, velocity matching
    gamma = 4e-4   # force 3, collision avoidance
    vmax  = 1.0    # force 4, maximum v limit

    # number of animals
    Na = 100

    # number of time points in simulation
    Nt = 50

    # time step
    dt = 1e-0
'''

    # reflecting boundary conditions?
    reflect = True

    # corral ranges
    xR = 100.0
    yR = 100.0
    zR = 100.0
```

```
lims = [xR, yR, zR]

# working arrays
r = np.zeros((Na,3))
v = np.zeros((Na,3))

#distance arrays
r2 = np.empty(shape=(Na,Na))

# initial conditions
for i in range(Na):
    for j in range(2):
        r[i,j] = 40 + 20*np.random.rand(1)
        v[i,j] = -0.5 + 1*np.random.rand(1)
    r[i,2] = zR / 2

plt.figure()
plt.plot(r[:,0], r[:,1], marker='o',linestyle='')
plt.xlim(0,xR)
plt.ylim(0,yR)
plt.gca().set_aspect('equal', adjustable='box')
#plt.axis('off')
plt.xticks([])
plt.yticks([])
plt.close()

filenm = dir + 'file_{:03d}.png'.format(0)
plt.savefig(filenm)

rcm = [0,0,0]

# force arrays
f1 = np.zeros((Na,3))
f2 = np.zeros((Na,3))
f3 = np.zeros((Na,3))

# helper arrays
rcmrel = np.zeros((Na,3))   # 4th element is the distance


for i in range(1,Nt):

    rcm, rcmrel = cm(r,Na)

    r2 = dist(r,Na)

    # calculate the cm force
    f1 = force1(alpha, rcmrel, Na)

    # calculate the velocity matching force
```

```
        f2 = force2(beta, r2, v, Na)

        # calculate the collision avoidance force
        f3 = force3(gamma, r, r2, Na)


        for j in range(Na):
            r[j,:] += v[j,:] * dt
            v[j,:] += (f1[j,:] + f2[j,:] + f3[j,:]) * dt


        # apply the speed limit
        v = force4(v, vmax, Na)

        if reflect == True:
            v = force5(r, v, lims, Na)

        print(i)

        plt.figure()
        plt.plot(r[:,0], r[:,1], marker='o',linestyle='')
        plt.xlim(0,xR)
        plt.ylim(0,yR)
        #plt.axis('off')
        plt.xticks([])
        plt.yticks([])
        plt.axis('off')

        filenm = dir + 'file_{:03d}.png'.format(i)
        plt.savefig(filenm)
        plt.close()

    #video(run)


def dist(r,n):

    r2 = np.zeros((n,n))

    for i in range(n-1):
        for j in range(i+1,n):
            r2[i,j] = np.sqrt( (r[i,0]-r[j,0])**2 + (r[i,1]-r[j,1])**2 + (r[i,2]-r[j,2])**2 )
            r2[j,i] = r2[i,j]

    return r2



def cm(r,Na):
```

```python
    rcm    = np.empty(3)
    rcmrel = np.empty(shape=(Na,3))

    # calculate the center of mass position
    for j in range(3):
        rcm[j] = np.sum(r[:,j])/Na

    for j in range(Na):
        rcmrel[j,0:3] = rcm - r[j,:]

    return rcm, rcmrel



def force1(a, r, n):

    f = np.zeros(shape=(n,3))

    for i in range(n):
        if np.sum(np.abs(r[i,:])) != 0:
            f[i,:] = a * r[i,:]

    return f


def force2(a, r2, v, n):

    f = np.zeros(shape=(n,3))

    for i in range(n):
        for j in range(n):
            for k in range(3):
                if i != j:
                    f[i,k] += a * np.sum(v[j,k] / r2[i,j])

    return f



def force3(a, r, r2, n):

    f = np.zeros(shape=(n,3))

    for i in range(n):
        for j in range(n):
            for k in range(3):
                if i != j:
                    f[i,k] += a * np.sum((r[i,k]-r[j,k]) / r2[i,j])

    return f
```

```
def force4(v, vm, n):

    for i in range(n):
        z = np.sum(v[i,:]**2)
        if z > vm**2:
            v[i,:] *= vm / np.sqrt(z)

    return v


def force5(r, v, lims, n):

    for i in range(n):

        for j in range(3):
            if (r[i,j] < 0) or (r[i,j] > lims[j]):
                v[i,j] *= -1

    return v



def video(run):

    image_folder = './' + run + '/'
    fps = 10

    image_files = [image_folder+'/'+img for img in sorted(os.listdir(image_folder)) if img.endswith(".png")]
    clip = moviepy.video.io.ImageSequenceClip.ImageSequenceClip(image_files, fps=fps)
    clip.write_videofile(run + '.mp4')
```