BOIDS CONTROL OF CHAOS

MAKOTO ITOH
Department of Information and Communication Engineering,
Fukuoka Institute of Technology
Fukuoka 811-0295, Japan

LEON O. CHUA
Department of Electrical Engineering and Computer Sciences,
University of California, Berkeley
Berkeley, CA 94720, USA

Received July 13, 2006; Revised October 18, 2006

Chaotic nonlinear networks are investigated, which are controlled by simple boids rules. They exhibit complex and emergent behaviors such as flocking behavior, separation behavior, joining behavior and obstacle avoiding behavior.

Keywords: Boids; collision avoidance; velocity matching; flock centering; obstacle avoidance; chaotic synchronization; cellular neural network.

1. Introduction
Boids (short for “Birdoid”) is an artificial life program, simulating animal motion such as flocking behavior of birds, herding behavior of land animals and schooling behavior of fish [Reynolds, 1987]. The dynamics of boids is described by discrete-time dynamical systems similar to cellular automata. As with most artificial life simulations, boids exhibit complex flocking behavior, which arises from the interaction of simple local rules. The three simple rules applied in the boids are as follows:

• Collision Avoidance: avoid collisions with nearby flockmates.
• Velocity Matching: attempt to match velocity with nearby flockmates.
• Flock Centering: attempt to stay close to nearby flockmates.

More complex rules can be added, such as obstacle avoidance and goal seeking. The characteristics of boids are described as follows:

• synchronize their speed with nearby flockmates;
• join when they meet;
• expand in a flash if they started too close together;
• aggregate slowly if they started too far apart.

A flock seems randomly arrayed and yet is magnificently synchronized. The interaction between simple behaviors of individuals produce complex yet organized group behavior. The behavior of boids can either be characterized as chaotic (splitting groups and wild behavior) or orderly. Unexpected behaviors, such as splitting flocks and reuniting after avoiding obstacles, can be considered emergent. The boids framework is often used in computer graphics, providing a realistic scene with a flock of birds, schools of fish or herds of animals (based on “boids,” Wikipedia: The Free Encyclopedia).

The boids work in a manner similar to cellular neural networks, since each boid acts autonomously and references a neighborhood, as cellular neural networks do [Chua, 1998; Chua & Roska, 2002]. The synchronization behavior of boids is also similar
to chaotic synchronization, since flocks behave in a chaotic fashion and synchronize their speed with nearby flockmates. Thus, there arises a question whether or not chaotic nonlinear networks or chaotic cellular networks controlled by the boids rules can exhibit complex and emergent behavior such as joining behavior and obstacle avoiding behaviors.

In this paper, we give an affirmative answer to this question. We first formulate a boids nonlinear network from the viewpoint of cellular neural networks. Then, we implement the following rules to the boids nonlinear network:

- move toward the average position of nearby flockmates;
- avoid collisions with nearby flockmates;
- match velocity with nearby flockmates;
- dodge a static obstacle;
- join to form a larger flock;
- keep a distance from different kinds of flock.

Some of these rules are implemented by using chaotic synchronization, and the remaining rules are implemented by switching over to other vector fields. Next, we investigate the complex flocking behavior and the emergent behavior by using computer simulations. Finally, some concluding remarks on controlling boids are given.

2. Boids Nonlinear Networks

Many nonlinear systems which are known to exhibit chaotic behavior are modeled by a set of nonlinear autonomous differential equations:

$$\frac{du_i}{dt} = f_i(u_1, u_2, \ldots, u_n) \quad (i = 1, 2, \ldots, n)$$  \hspace{1cm} (1)

where \( u = (u_1, u_2, \ldots, u_n) \) are state variables, and \( f(u) = (f_1(u), f_2(u), \ldots, f_n(u)) \) is a nonlinear vector function of \( u \).

Given initial state \( u_i^\alpha(0) \) at \( t = 0 \), the state \( u_i^\alpha \) of each isolated boid \( B_{\alpha} \) is assumed to evolve for all \( t \geq 0 \) via state equations:

$$\frac{du_i^\alpha}{dt} = f_i(u_1^\alpha, u_2^\alpha, \ldots, u_n^\alpha) \quad (i = 1, 2, \ldots, n)$$  \hspace{1cm} (2)

We will assume for simplicity that all boids are identical and each boid is coupled locally only to those neighbor boids whose trajectories lie inside a pre-scribed sphere \( S_\alpha \) of radius \( \epsilon \):

$$S_\alpha(\epsilon, t) = \left\{ B_\beta : r_{\alpha, \beta} = \sqrt{\sum_{i=1}^{n} (u_i^\alpha(t) - u_i^\beta(t))^2} \leq \epsilon \right\}.$$  \hspace{1cm} (3)

at time \( t \), where \( r_{\alpha, \beta} \) indicates the distance between the boids \( B_\alpha \) and \( B_\beta \). We will usually delete \( \epsilon \) and \( t \) from \( S_\alpha(\epsilon, t) \) and simply write \( S_\alpha \) to avoid clutter. Then, the dynamics of the locally coupled chaotic nonlinear networks, namely, the dynamics of boids nonlinear networks is defined by

$$\frac{du_i^\alpha}{dt} = f_i(u_1^\alpha, u_2^\alpha, \ldots, u_n^\alpha) + \sum_{B_\beta \in S_\alpha} D_i^\beta g_i(u_1^\beta, u_2^\beta, \ldots, u_n^\beta)$$  \hspace{1cm} (4)

where \( D_i^\beta(i = 1, 2, \ldots, n) \) are coupling coefficients, and \( g(u) = (g_1(u), g_2(u), \ldots, g_n(u)) \) is a nonlinear vector function of \( u \).

The dynamics of Eq. (4) is quite similar to the dynamics of cellular neural networks [Chua, 1998; Chua & Roska, 2002]. However, in the boids nonlinear networks, the number of boids belonging to \( S_\alpha \) may change continuously as time \( t \) increases. For example, unexpected behaviors, such as splitting flocks and reuniting after avoiding obstacles cause changes in the number of boids belonging to \( S_\alpha \).

3. Implementation of Basic Boids Rules

In this section, we illustrate the implementation of several simple boids rules to Eq. (4) by using chaotic synchronization and by switching over to other vector fields.

Rule 1. Flock Centering: Boids attempt to move toward the average position of nearby flockmates.

The center of nearby flockmates is defined by

$$\overline{u_i^\alpha}(t) = \frac{\sum_{\beta \in S_\alpha} u_i^\beta(t)}{N_\alpha},$$  \hspace{1cm} (5)
where $N_\alpha$ indicates the number of nearby flockmates. The boids can move toward the center $u_i^{\alpha}$ by using chaotic synchronization [Itoh & Chua, 2004]. Thus, flock centering is implemented here by imposing the control dynamics

$$
\frac{du_i^{\alpha}}{dt} = f_i(u_1^{\alpha}, u_2^{\alpha}, \ldots, u_n^{\alpha}) + d_i^{\alpha}(u_i^{\alpha} - u_i^{\alpha}).
$$

(6)

where $d_i^{\alpha} > 0$.

Rule 2. Collision Avoidance: Boids attempt to avoid collisions with nearby flockmates.

If two flockmates get close to each other, boids must attempt to avoid collisions. In the case where the distance $r_{\alpha,\beta}$ between boids $B_\alpha$ and $B_\beta$ becomes less than $\delta > 0$, the boids can scatter from the center of nearby flockmates by chaotic desynchronization [Itoh & Chua, 2004]. Thus, collision avoidance can be implemented by imposing the dynamics

$$
\begin{align*}
\frac{du_i^{\alpha}}{dt} &= f_i(u_1^{\alpha}, u_2^{\alpha}, \ldots, u_n^{\alpha}) + e_i^{\alpha}(u_i^{\alpha} - u_i^{\alpha}), \\
\frac{du_i^{\beta}}{dt} &= f_i(u_1^{\beta}, u_2^{\beta}, \ldots, u_n^{\beta}) + e_i^{\beta}(u_i^{\beta} - u_i^{\beta}).
\end{align*}
$$

(7)

where $e_i^{\alpha} \leq 0$, $e_i^{\beta} \leq 0$.

Rule 3. Velocity Matching: Boids attempt to match velocity with nearby flockmates.

The average velocity of nearby flockmates is defined by

$$
\bar{f_i^{\alpha}} = \frac{\sum_{\beta \in S_\alpha} f_i(u_1^{\beta}, u_2^{\beta}, \ldots, u_n^{\beta})}{N_\alpha}.
$$

(8)

If the velocity of boids is too large or too small compared to the average velocity of the flockmates, namely,

$$
\sqrt{\frac{\sum_{i=1}^{N} f_i(u_1^{\alpha}, u_2^{\alpha}, \ldots, u_n^{\alpha})}{\sqrt{\sum_{i=1}^{N} (\bar{f_i^{\alpha}})^2}}} \geq R_{\text{max}},
$$

(9)

or

$$
\frac{\sum_{i=1}^{N} f_i(u_1^{\alpha}, u_2^{\alpha}, \ldots, u_n^{\alpha})}{\sqrt{\sum_{i=1}^{N} (f_i^{\alpha})^2}} \geq R_{\text{max}},
$$

(10)

where $0 < R_{\text{min}} < 1$ and $R_{\text{max}} > 1$. then velocity matching can be implemented by switching over to a new vector field defined by:

$$
\frac{du_i^{\alpha}}{dt} = (1 - \lambda)f_i(u_1^{\alpha}, u_2^{\alpha}, \ldots, u_n^{\alpha}) + \lambda\bar{f_i^{\alpha}}
$$

$$
= f_i(u_1^{\alpha}, u_2^{\alpha}, \ldots, u_n^{\alpha}) + \lambda(\bar{f_i^{\alpha}} - f_i(u_1^{\alpha}, u_2^{\alpha}, \ldots, u_n^{\alpha}))
$$

(11)

where $0 \leq \lambda \leq 1$.

The dynamics of Eq. (11) can be continuously deformed from $f_i(u_1^{\alpha}, u_2^{\alpha}, \ldots, u_n^{\alpha})$ to the average vector field $\bar{f_i^{\alpha}}$ by increasing $\lambda$ from 0 to 1 (such a deformation is called a homotopy between the two functions). Note that when $\lambda \neq 0$ and $\lambda \neq 1$, the dynamics of Eq. (11) tends to the weighted average of $f_i(u_1^{\alpha}, u_2^{\alpha}, \ldots, u_n^{\alpha})$ and $\bar{f_i^{\alpha}}$. In this paper, we set $\lambda = 1$ for the sake of simplicity. Thus, the current vector field is switched over to the average vector field of nearby flockmates:

$$
\frac{du_i^{\alpha}}{dt} = \bar{f_i^{\alpha}}.
$$

(12)

4. Implementation of Other Boids Rules

In this section, we illustrate the implementation of several other important boids rules.

Rule A. Obstacle Avoidance: Boids attempt to dodge static obstacles.

Assume that a static obstacle is defined by the equation

$$
h(u_1, u_2, \ldots, u_n) = b \quad (i = 1, 2, \ldots, n)
$$

(13)

where $h$ is a scalar function of $u = (u_1, u_2, \ldots, u_n)$ and $b$ is a constant. The normal vector at $u = (u_1, u_2, \ldots, u_n)$ on a surface $h(u_1, u_2, \ldots, u_n) = b$ is given by

$$
\nabla h(u_1, u_2, \ldots, u_n) \equiv \left( \frac{\partial h(u_1, u_2, \ldots, u_n)}{\partial u_1}, \frac{\partial h(u_1, u_2, \ldots, u_n)}{\partial u_2}, \ldots, \frac{\partial h(u_1, u_2, \ldots, u_n)}{\partial u_n} \right).
$$

(14)
If a boid gets close enough to a static obstacle, that is, if the distance between a boid and a static obstacle is less than $\epsilon_o$, the boid must attempt to dodge the static obstacle. Obstacle avoidance can be implemented by switching over to a new vector field:

$$\frac{du_i^o}{dt} = (1 - \mu_i)f_i(u_1^o, u_2^o, \ldots, u_n^o)$$
$$+ \mu_i \gamma \frac{\partial h(u_1^o, u_2^o, \ldots, u_n^o)}{\partial u_i}$$
$$= f_i(u_1^o, u_2^o, \ldots, u_n^o)$$
$$+ \mu_i \left( \gamma \frac{\partial h(u_1^o, u_2^o, \ldots, u_n^o)}{\partial u_i} - f_i(u_1^o, u_2^o, \ldots, u_n^o) \right),$$  

(15)

where $0 \leq \mu_i \leq 1$ and $\gamma > 0$. In this paper, we set $\mu_i = 1$ for the sake of simplicity.

**Rule B. Joining a Flock: Boids of the same feather flock together.**

Flocks of the same kind (or of the same group) may attempt to join together to become larger flocks if the distance between the centers of two flocks becomes less than $\epsilon_g > 0$. Joining of the flock can be implemented by the dynamics of chaotic synchronization

$$\frac{du_i^o}{dt} = f_i(u_1^o, u_2^o, \ldots, u_n^o) + w_i^o(\overline{u_i} - u_i^o),$$  

(16)

where $w_i^o > 0$ and $\overline{u_i}$ indicates the center of the united the flocks.

**Rule C. Separation of Flocks: Boids keep a distance from different kinds of flocks.**

A flock may attempt to go away from other kinds of flocks. If a flock gets close enough to a different kind of flocks (or different groups of flocks), that is, if the distance between the centers of two flocks becomes less than $\epsilon_g > 0$, boids attempt to scatter. Separation of flocks is implemented by the dynamics of chaotic desynchronization

$$\frac{du_i^o}{dt} = f_i(u_1^o, u_2^o, \ldots, u_n^o) + s_i^o(\overline{u_i}^o - u_i^o),$$  

(17)

where $s_i^o < 0$ and $\overline{u_i}^o$ indicates a center of nearby flockmates.

5. **Priority of Rules**

A boid can have conflicting requests since flocking behavior is simply the result of the interaction of simple rules. For example, if a wall is directly ahead and if the flock centering request and the obstacle avoidance request are in an opposite and therefore canceling directions, the boid might make only a small turn and crash into the wall. The highest-priority must be given to obstacle avoidance. Thus, the behavior of boids is modeled by using a priority-based scheme:

<table>
<thead>
<tr>
<th>priority</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>high</td>
<td>obstacle avoidance, collision avoidance, separation</td>
</tr>
<tr>
<td>low</td>
<td>joining, centering, velocity matching</td>
</tr>
</tbody>
</table>

In a priority ordering method [Reynolds, 1987], each of the behavioral urges is expressed as an acceleration request or vector. The acceleration requests are ranked in priority order and added into an accumulator of the boids simulation model. The magnitude of each request is measured and added into another accumulator. This continues until the sum of the accumulated magnitudes gets larger than the maximum acceleration value, and then no more requests are taken. In an emergency the acceleration would be allocated to satisfy the most pressing needs first; if all available acceleration is “used up,” the less pressing behaviors might be temporarily unsatisfied. For example, a flock centering request could be ignored if there were a strong collision avoidance request at the same time [Reynolds, 1987]. This paper uses the simplified form of the priority ordering method. That is, if there are some acceleration requests, a boid chooses only one behavior at a time according to the following priority order list:

<table>
<thead>
<tr>
<th>priority</th>
<th>order</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>high</td>
<td>1</td>
<td>obstacle avoidance</td>
</tr>
<tr>
<td>↓</td>
<td>2</td>
<td>collision avoidance</td>
</tr>
<tr>
<td>↓</td>
<td>3</td>
<td>separation</td>
</tr>
<tr>
<td>↓</td>
<td>4</td>
<td>joining</td>
</tr>
<tr>
<td>↓</td>
<td>5</td>
<td>centering</td>
</tr>
<tr>
<td>low</td>
<td>6</td>
<td>velocity matching</td>
</tr>
</tbody>
</table>

In place of the accumulators, the priority order list is used to decide the behavior.
6. Computer Simulations of Boids Behavior

It is well known that Chua's oscillators [Madan, 1993] and Lorenz systems [Lorenz, 1963] exhibit chaotic behaviors for some parameter regions. Their dynamics are defined as follows:

- **Chua's oscillator** [Madan, 1993]

\[
\begin{align*}
\frac{dx}{dt} &= \alpha(y - x - g(x)), \\
\frac{dy}{dt} &= x - y + z, \\
\frac{dz}{dt} &= \beta y - \gamma z,
\end{align*}
\]  

(18)

where \(\alpha, \beta\) and \(\gamma\) are constants and

\[g(x) = bx + \frac{1}{2}(a - b)(|x + 1.0| - |x - 1.0|).\]  

(19)

Chua's oscillator (18) has chaotic attractors for \(\alpha = 10, \beta = 10.5, \gamma = 0.45, a = -1.22, b = -0.734\).

- **Lorenz system** [Lorenz, 1963]

\[
\begin{align*}
\frac{dx}{dt} &= \sigma(y - x), \\
\frac{dy}{dt} &= -xz + rx - y, \\
\frac{dz}{dt} &= xy - bz,
\end{align*}
\]  

(20)

where \(\sigma, r\) and \(b\) are constants. The Lorenz system (20) has a chaotic attractor for \(\sigma = 10, b = 8/3, r = 28\).

We show next some computer simulations of complex flocking behaviors by using Chua's oscillators and Lorenz systems.

6.1. Flocking behavior

The computer simulations of flocking behavior of two boids are given in Figs. 1–6. The trajectories between two boids are illustrated in Figs. 1 and 2. Observe that they move flocking together. The distance between two boids is given in Figs. 3 and 4. Observe that the two boids maintain a small distance \(\delta \approx 0.1\) in the case of Chua's oscillators, and \(\delta \approx 0.5\) in the case of the Lorenz systems. The synchronization behavior of two boids is given in Figs. 5 and 6. Observe that the two boids do not synchronize completely, and therefore their Lissajous figures in Figs. 5 and 6 exhibit some thickness. A thin line would imply a perfect synchronization, resulting possibly in collisions. In these computer simulations, the following parameters are used:

<table>
<thead>
<tr>
<th>Chua's oscillator</th>
<th>parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>flock centering</td>
<td>(d_1^\alpha = 10)</td>
</tr>
<tr>
<td>collision avoidance</td>
<td>(e_1^\alpha = -5, \delta = 0.1)</td>
</tr>
<tr>
<td>velocity matching</td>
<td>(R_{\text{max}} = 2.0, R_{\text{min}} = 0.2)</td>
</tr>
</tbody>
</table>

![Fig. 1. Flocking of two boids (Chua's oscillators).](attachment:image.png)
Fig. 2. Flocking of two boids (Lorenz system).

Fig. 3. Distance between two boids (Chua's oscillators).

<table>
<thead>
<tr>
<th>Lorenz system</th>
<th>rule</th>
<th>parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>flock centering</td>
<td>$d^{i}_{t} = 20$</td>
<td></td>
</tr>
<tr>
<td>collision avoidance</td>
<td>$e^{i}_{t} = -10$, $\delta = 0.5$</td>
<td></td>
</tr>
<tr>
<td>velocity matching</td>
<td>$R_{\text{max}} = 1.5$, $R_{\text{min}} = 0.3$</td>
<td></td>
</tr>
</tbody>
</table>

6.2. Separation of flocks

The computer simulations of separation behavior of two flocks are illustrated in Figs. 7–12. The trajectories of two flocks are given in Figs. 7 and 8. Since the distance among a small subset of blue trajectories are very small and all trajectories are superimposed over the same region in space, we can see only red
Fig. 4. Distance between two boids (Lorenz system).

Fig. 5. Synchronization of two boids (Chua’s oscillators).
Fig. 6. Synchronization of two boids (Lorenz system).

Fig. 7. Separation of two flocks (Chua’s oscillator). The trajectories in green and in light blue are hidden in the trajectory in blue, since our plotting software GNUPLLOT graphs the trajectories in order of green, light blue, blue. Similarly, the trajectories in orange and in pink are hidden in the trajectory in red, since GNUPLLOT graphs the trajectories in order of orange, pink, red.
Fig. 8. Separation of two flocks (Lorenz system). The trajectories in green and in light blue are hidden in the trajectory in blue, since GNUPLOT graphs the trajectories in order of green, light blue, blue. Similarly, the trajectories in orange and in pink are hidden in the trajectory in red, since GNUPLOT graphs the trajectories in order of orange, pink, red.

Fig. 9. Distance between two flocks (Chua's oscillator).
Fig. 10. Distance between two flocks (Lorenz system).

Fig. 11. Desynchronization of two boids (Chua's oscillators #1a and #2a).
and blue lines.\(^2\) The symbol \(\#ip\) indicates the \(p\)th boid of each flock (group) \(\#i\) \((i = 1, 2, p = a, b, c)\). In Figs. 7 and 8, there are two flocks, each composed of three boids \(a, b\) and \(c\). Hence, \(\#ib\) denotes boid \(b\) from flock 1, and \(\#ic\) denotes boid \(c\) from flock 2. The distance between two flocks is given in Figs. 9 and 10. Observe that each flock maintains a distance from the other flock. The synchronization behavior is given in Figs. 11 and 12. Observe that the two flocks belong to different kinds of flocks do not synchronize, since the flock attempts to go away from other kinds of flocks. In these computer simulations, the following parameters are used:

### Chua’s oscillator

<table>
<thead>
<tr>
<th>rule</th>
<th>parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>flock centering</td>
<td>(d_i^\alpha = 15)</td>
</tr>
<tr>
<td>collision avoidance</td>
<td>(e_i^\alpha = -0.1, \delta = 0.02)</td>
</tr>
<tr>
<td>velocity matching</td>
<td>(R_{\text{max}} = 1.5, R_{\text{min}} = 0.1)</td>
</tr>
<tr>
<td>separation of flocks</td>
<td>(\epsilon_g = 3, s_i^\alpha = -0.1)</td>
</tr>
</tbody>
</table>

### Lorenz system

<table>
<thead>
<tr>
<th>rule</th>
<th>parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>flock centering</td>
<td>(d_i^\alpha = 15)</td>
</tr>
<tr>
<td>collision avoidance</td>
<td>(e_i^\alpha = -1.0, \delta = 0.05)</td>
</tr>
<tr>
<td>velocity matching</td>
<td>(R_{\text{max}} = 2.0, R_{\text{min}} = 0.1)</td>
</tr>
<tr>
<td>separation of flocks</td>
<td>(\epsilon_g = 3, s_i^\alpha = -5)</td>
</tr>
</tbody>
</table>

6.3. Joining a flock

The computer simulations of the “joining” behavior are illustrated in Figs. 13–16. Observe that two flocks join to become a larger flock as time increases. In Fig. 13, a small subset of blue trajectories for Chua’s oscillator is initially quite apart from a small subset of red trajectories. In Fig. 14, they converge toward each other. Since all trajectories are superimposed over the same region in space, we can see only a red line after the two flocks are joined.\(^3\) Similarly, in Fig. 15, a small subset of blue trajectories for the Lorenz system is initially apart from a small

---

\(^2\) When our plotting software GNUPLOT graphs multiple lines simultaneously, this overwrites the existing plot, and consequently the existing trajectories are hidden behind the new trajectories. In Fig. 7, the trajectories in green and light blue are hidden in the trajectory in blue, since GNUPLOT graphs the trajectories in order of green, light blue, blue. Similarly, the trajectories in orange and in pink are hidden in the trajectory in red, since GNUPLOT graphs the trajectories in order of orange, pink, red.

\(^3\) GNUPLOT overwrites the existing plot, and it graphs the trajectories in order of green, light blue, orange, pink, red. Therefore, we can see only a red line after the two flocks are joined.
Fig. 13. Joining a flock (Chua's oscillators) for $t \in [0, 50]$. GNUPLOT overwrites the existing plot, and it graphs the trajectories in order of green, light blue, blue, orange, pink, red. Therefore, we can see only a red line after the two flocks are joined.

Fig. 14. Joining a flock (Chua's oscillators) for $t \in [150, 200]$. GNUPLOT overwrites the existing plot, and it graphs the trajectories in order of green, light blue, blue, orange, pink, red. Therefore, we can see only a red line after the two flocks are joined.
Fig. 15. Joining a flock (Lorenz systems) for $t \in [0, 40]$. GNUPLOT overwrites the existing plot, and it graphs the trajectories in order of green, light blue, blue, orange, pink, red. Therefore, we can see only a red line after the boids joined a flock.

Fig. 16. Joining a flock (Lorenz systems) for $t \in [90, 100]$. GNUPLOT overwrites the existing plot, and it graphs the trajectories in order of green, light blue, blue, orange, pink, red. Therefore, we can see only a red line after the boids joined a flock.
subset of red trajectories. In Fig. 16, they converge toward each other. In these computer simulations, we used the following parameters:

**Chua’s oscillator**

<table>
<thead>
<tr>
<th>rule</th>
<th>parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>flock centering</td>
<td>( d_i^\alpha = 15 )</td>
</tr>
<tr>
<td>collision avoidance</td>
<td>( e_i^\alpha = -0.2, \delta = 0.01 )</td>
</tr>
<tr>
<td>velocity matching</td>
<td>( R_{\text{max}} = 2.0, R_{\text{min}} = 0.1 )</td>
</tr>
<tr>
<td>joining a flock</td>
<td>( \epsilon_g = 5, w_i^\alpha = 15 )</td>
</tr>
</tbody>
</table>

**Lorenz system**

<table>
<thead>
<tr>
<th>rule</th>
<th>parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>flock centering</td>
<td>( d_i^\alpha = 15 )</td>
</tr>
<tr>
<td>collision avoidance</td>
<td>( e_i^\alpha = -1.0, \delta = 0.05 )</td>
</tr>
<tr>
<td>velocity matching</td>
<td>( R_{\text{max}} = 2.0, R_{\text{min}} = 0.1 )</td>
</tr>
<tr>
<td>joining a flock</td>
<td>( \epsilon_g = 10, w_i^\alpha = 15 )</td>
</tr>
</tbody>
</table>

6.4. **Obstacle avoidance**

Let us define a sphere of radius \( r_1 \) centered at \((x_1, y_1, z_1)\) by

\[
(x - x_1)^2 + (y - y_1)^2 + (z - z_1)^2 = r_1^2,
\]  

and its normal vector \( \mathbf{n} = (n_x, n_y, n_z) \) at the point \((x, y, z)\) by

\[
(n_x, n_y, n_z) = \left(2(x-x_1), 2(y-y_1), 2(z-z_1)\right).
\]  

Let us define a cylinder of radius \( r_2 \) centered at \((x_2, y_2)\) by

\[
(x - x_2)^2 + (y - y_2)^2 = r_2^2,
\]  

and its normal vector \( \mathbf{n} = (n_x, n_y, n_z) \) at the point \((x, y, z)\) by

\[
(n_x, n_y, n_z) = \left(2(x-x_1), 2(y-y_1), 0\right).
\]  

Thus, the “sphere” and the “cylinder” obstacles are specified by the parameters:

\[
(x_1, y_1, z_1, r_1),
\]  

and

\[
(x_2, y_2, r_2),
\]  

respectively.

Our computer simulations of the obstacle (a sphere and a cylinder) avoidance behavior are illustrated in Figs. 17–22. Observe that the flock dodges the static obstacle (sphere in blue) located on its normal trajectory path. Since all trajectories are superimposed over the same region in space, we can see a red line.\(^4\) In Fig. 17, the flock of Chua’s

\[\text{Fig. 17. Obstacle avoidance (Chua's oscillators). GNUPLOT overwrites the existing plot, and it graphs the trajectories in order of green, light blue, orange, pink, red. Therefore, we can see only a red line after the boids joined a flock.}\]

\[\text{\(^4\)GNUPLOT overwrites the existing plot, and it graphs the trajectories in order of green, light blue, orange, pink, red. Therefore, we can see only a red line after the boids joined a flock.}\]
oscillators avoids the sphere by changing its current direction continuously. In Fig. 18, the flock of Lorenz systems also dodges the sphere by moving upward. In Fig. 19, the flock of Chua's oscillators avoids the cylinder (in blue) by getting around it.

In Fig. 20, the flock of Lorenz systems avoids the cylinder by climbing up its wall. In Figs. 21 and 22, the view from the z-axis is shown.

In our computer simulations, the following parameters are chosen for avoiding collision with
Fig. 20. Obstacle avoidance (Lorenz systems). GNUPLOT overwrites the existing plot, and it graphs the trajectories in order of green, light blue, orange, pink, red. Therefore, we can see only a red line after the boids joined a flock.

Fig. 21. Obstacle avoidance on (x, y)-plane (Chua’s oscillators). GNUPLOT overwrites the existing plot, and it graphs the trajectories in order of green, light blue, orange, pink, red. Therefore, we can see only a red line after the boids joined a flock.
the "sphere" obstacle:

<table>
<thead>
<tr>
<th>Chua's oscillator</th>
<th>parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>flock centering</td>
<td>$d_i^\alpha = 12$</td>
</tr>
<tr>
<td>collision avoidance</td>
<td>$e_i^\alpha = -0.1, \delta = 0.02$</td>
</tr>
<tr>
<td>velocity matching</td>
<td>$R_{\text{max}} = 4.0, R_{\text{min}} = 0.2$</td>
</tr>
<tr>
<td>obstacle avoidance</td>
<td>$\mu_1 = 1, \mu_2 = 1, \mu_3 = 1$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>obstacle</th>
<th>parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>sphere</td>
<td>$x_1 = 3.5, y_1 = 0, z_1 = -3.5, r_1 = 0.6$</td>
</tr>
</tbody>
</table>

The following parameters\(^5\) are used for avoiding collision with the "cylinder" obstacle:

<table>
<thead>
<tr>
<th>Chua's oscillator</th>
<th>parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>flock centering</td>
<td>$d_i^\alpha = 5$</td>
</tr>
<tr>
<td>collision avoidance</td>
<td>$e_i^\alpha = -0.2, \delta = 0.02$</td>
</tr>
<tr>
<td>velocity matching</td>
<td>$R_{\text{max}} = 5.0, R_{\text{min}} = 0.1$</td>
</tr>
<tr>
<td>obstacle avoidance</td>
<td>$\epsilon_0 = 0.2, \gamma = 0.1$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>obstacle</th>
<th>parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>cylinder</td>
<td>$x_2 = 3, y_2 = 0, r_1 = 0.15$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Lorenz system</th>
<th>parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>flock centering</td>
<td>$d_i^\alpha = 12$</td>
</tr>
<tr>
<td>collision avoidance</td>
<td>$e_i^\alpha = -0.1, \delta = 0.02$</td>
</tr>
<tr>
<td>velocity matching</td>
<td>$R_{\text{max}} = 5, R_{\text{min}} = 0.1$</td>
</tr>
<tr>
<td>obstacle avoidance</td>
<td>$\epsilon_0 = 5, \gamma = 0.1$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>obstacle</th>
<th>parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>cylinder</td>
<td>$x_2 = 15, y_2 = 10, r_2 = 4.5$</td>
</tr>
</tbody>
</table>

\(^5\)Note that we set $\mu_3 = 0$ since $n_z \equiv 0$ [see Eq. (15)].
7. Concluding Remarks

We have investigated several chaotic nonlinear networks controlled by several boids rules. They exhibited complex and emergent behaviors. The "synchronization" phenomenon can only be achieved with the proposed model. In this paper, the chaotic boids are controlled by using three state variables, and all boids are assumed to be identical for simplicity. Therefore, there still remain several problems:

- control of boids by using fewer state variables;
- control of nonidentical boids or nonautonomous boids.

Furthermore, the following interesting rules are not implemented:

- goal seeking rule;
- predator–prey interaction rule;
- leader following rule.

It is not difficult to realize them by improving and refining the models in this paper. Furthermore, many CNN image processing techniques are available to their implementation.

Acknowledgment

This paper is supported in part by the MURI contract no. N00014-03-1-0698.

References


