Flocking Cars: Dynamic Behavior in Simulation & Implementation

Alex T Jacobs, Jeffrey H Koessler, *University of Arizona, ECE 516 Introduction to Robotics*

***Abstract*—A simulation model and hardware implementation for studying vehicle flocking behavior is proposed. The simulation model consists of a system of n > 2 vehicles capable of zero turn radius maneuvering following the three basic flocking principles of long distance attraction, short distance repulsion, and heading alignment. The hardware implementation consists of wirelessly controlled cars with Ackerman steering mechanisms modified to use standard servos and the addition of a custom control board on each vehicle integrating XBee serial communication to host computer. A Vicon vision system with 8 cameras is utilized to provide vehicle position and orientation states. Within the context of the vision system physical capture area limitations, the hardware implementation demonstrates good agreement with the simulation model.**

I. Introduction

FLOCKINGIREFLY synchrony behavior has been the focus of many journal articles across a spectrum of disciplines, with most citing Reynolds (1986) as the genesis of codified examples. A graphical simulation of a group of "boids" was generated that followed the now ubiquitous three basic rules of flocking:

a) Long distance **Cohesion** to achieve clustering
b) Short distance **Separation** to avoid collisions
c) Heading **Alignment** for all to remain in formation

A thorough mathematical proof of convergence is presented by Tanner *et al.* [1] for the case of a time invariant agent-to-agent connection model, or fixed topology. A key feature of this formulation is the use of an artificial potential energy function to describe the repulsive and attractive forces, a combination of the basic rules a) and b) above. An example artificial potential function is presented as:

$$V_{ij}(\|r_{ij}\|) = \frac{1}{\|r_{ij}\|^2} + log\|r_{ij}\|^2, \qquad (1)$$

where r_{ij} is the relative distance between two agents. This potential function is plotted in **Figure 1**. Although not presented as such in [1], it is evident that the slope of such a curve represents the magnitude and sign of the force between two agents. This force is zero when the potential function is the lowest (i.e. the two agents are at the arbitrarily established "ideal" separation). It has a steep (tending to infinity) negative or repulsive force when the distance between the agents is less than the ideal, and a shallower positive or attractive force when the distance is greater than ideal. The rigorous proof by Tanner *et al.* achieves an intuitive result: the steady state of the agents is a configuration which minimizes the total system potential energy. The claim is made that this is true regardless of the system initial condition.

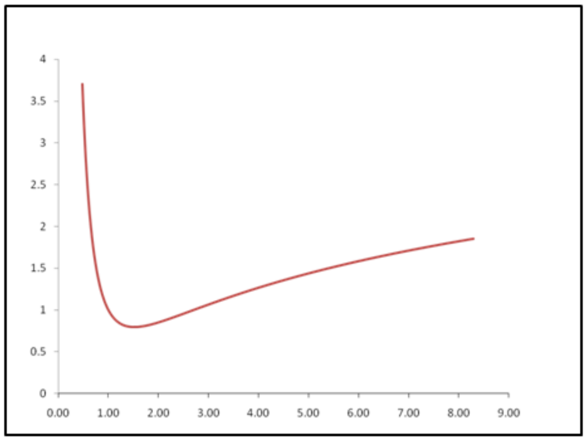


Figure 1: Tanner *et al.* [1] artificial potential function

Tanner *et al.* [2] expands on the results of the previous work to include dynamic topology, where the kinematics of the agents has an influence on the connectedness of the agents. With a slightly altered artificial potential energy function that introduces a non-smooth transition to a constant value, [2] presents the resulting agent configuration that again minimizes the total system potential energy.

A discussion and formulation on the geometric structure within a group of agents exhibiting flocking behavior is presented by Olfati-Saber [3]. Although beyond the scope of this present work, it is nevertheless worthwhile to consider the varying spatial organization of agents within the flock.

Flocking behavior was the subject of a TED talk hosted by S. Strogatz [4]. While the technical discussion is muted given the wide range of disciplines in the audience, there are vivid movie clips presented that reinforce the natural origin of flocking behavior in birds, fish, and other animals including humans. In addition [4] introduces the concept of a flocking "destabilizer" in the form of a natural predator with very strong repulsive force.
exemplified by the Buck and Buck series beginning in 1937 and including the 1976 paper [1] that documents a range of field observations and laboratory experiments. These experiments led the authors to construct a schematic model of a resetting timing

The rest of the paper is organized as follows: Section II presents the flocking algorithm. An overview of the flocking simulation is provided in Section III. In Section IV the hardware implementation is presented, along with a description of the Hybrid Dynamics Laboratory where the hardware demonstrations were performed. A concluding summary and ideas for future work are given in Section V.

II. Flocking Algorithm

The algorithm for flocking is relatively simple. For any car of index i, there is a desired direction vector $\vec{D_i}$. This vector points in the direction which the car should go to maintain the appropriate distance from its neighbors. At a given time step,

$$\vec{D_i} = \sum_{j \neq i} \vec{\sigma}(\vec{d_j}), \tag{2}$$

where $\vec{d_j}$ is the distance vector from the i^{th} to the j^{th} car, and $\vec{\sigma}(\,)$ is the vector squashing function:

$$\vec{\sigma}(\vec{v}) = \beta \frac{\vec{v}}{|\vec{v}|}\left(-C + \frac{|\vec{v}|}{|\vec{v}| + \gamma}\right) \tag{3}$$

This function returns the null vector when $\overrightarrow{|v|} = \frac{C\gamma}{1-C}$. β can be changed to change the scale of the dynamics. A plot of this function for $\beta = 2.0$, $C = 0.7$, and $\gamma = 0.214$ is shown in **Figure 2**. In this case, for two cars that are 0.5 m away, $\vec{\sigma}(\,)$ returns the null vector, and the cars are not affected by each other. For cars less than 0.5 m away, their contribution to the other cars desired vector is their distance vector renormalized with the sign changed. So for cars less than 0.5 m apart, the contribution to the desired vector is repulsive. Analogously, cars greater than 0.5 m apart have an attractive desired direction contribution.

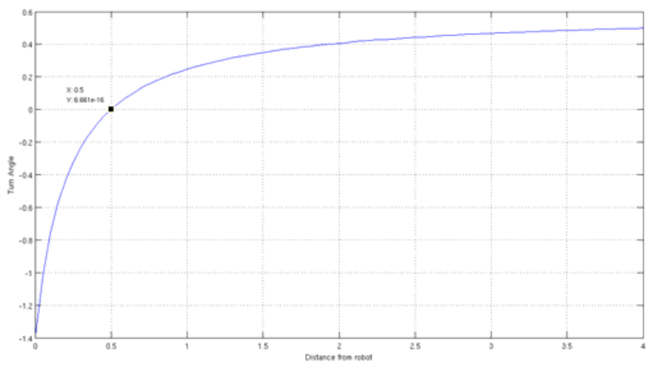


Figure 2: Squashing function. The input vector is normalized to a new value according to this graph.

Another important note about $\vec{\sigma}(\,)$ is that it "squashes" vectors that are very large ($\lim_{|\vec{v}|\to\infty} |\vec{\sigma}(|\vec{v}|)| = \beta(1 - C)$), while it gets very large in magnitude and negative as cars get close. This is required for the desired dynamics—as cars get close they should repel from each other with highest priority, and as cars distance themselves greatly they should be noticed, but with low priority. If $\vec{\sigma}(\,)$ instead continued increasing say linearly with $|\vec{v}|$, cars would run into closer cars on their heading toward a distanced car.

The next step in the algorithm is to use this desired vector to control the Ackermann-like steering of the each car to head to the desired direction. Note that the steering is not true Ackermann steering. Rather the wheels have the same angle with respect to the cars' orientation. To map $\vec{D}$ onto the steering of the i^{th} car, the wheel angle was adjusted to point wheels in the direction of $\vec{D_i}$. If this angle was unattainable (max wheel angles were roughly ±45°), the wheel angle was set the angle which most closely oriented the wheels toward $\vec{D}$ (similar to taking the floor or ceiling of a function).

Initially the throttle for the cars is was simply set to full throttle. In situ this caused a given car to slip and spin out if the car was turning too sharply, so the throttle instead was set to $max - \alpha \cdot Abs(\theta)$, where θ is the turning angle, *max* is the maximum value, and *α* is a scaling factor.

III. Simulation

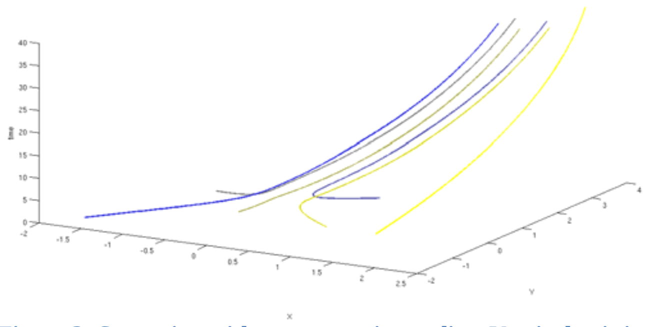


Figure 3: Swarming with a zero turning radius. Vertical axis is time.

A. Matlab simulation

The first simulation presented was written in Matlab, and a sample run is pictured in **Figure 3**. The simulation involves no Ackermann dynamics, but rather at each time step a car is displaced by some delta proportionate to $\vec{D}$. There is an extra car, the position of which is controlled precisely in code. This extra car is used to control the position of the rest of the cars.

This simulation is very unrealistic for arbitrary initial conditions. However, in situ if cars begin with the same initial speed and orientation, and they follow a target that has the same velocity as the cars maximums velocity, the dynamics are quite similar.

B. Flash simulation

The second simulation (pictured in **Figure 4** on the following page) uses Ackermann dynamics and is written in Flash so the user can interact with the cars. The mouse location is presented in the program as an additional car's location. The cars will therefore attempt group with each other and the mouse as well. While this simulation shows the desired behavior and incorporates Ackermann steering, it is unrealistic in that the cars do not slip and have a very small turning radius. The latter allows the cars to have no overall heading (meaning extra car is not moving), but still manage few collisions.

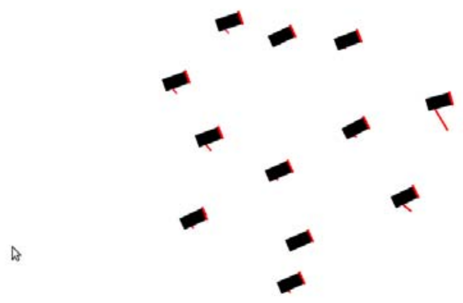


Figure 4: Basic Ackermann dynamics in an interactive Flash simulation

IV. Hardware Implementation

The overall setup is pictured in **Figure 5**. A central computer running Matlab receives positional data from the Vicon camera system. Using this information the central computer issues commands to each car using a wireless XBee network. Each car has a custom board including an AVR microcontroller, programmed with AVR-libc.

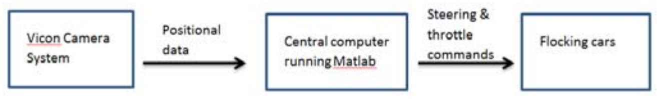


Figure 5: Overall setup

A. Wireless

The XBee modules have two modes called 'AT' and 'API' modes.

AT mode is intended to be used for point to point communication. A pair of XBees in AT mode are often referred to as a 'wireless serial port,' because each XBee repeats what is sent to them to a target XBee. The target XBee is determined via a lengthy pre-programming.

API mode is intended to be used with an API. This mode gives the user access to all XBee functions (analog/digital I/O, etc). One important feature of API mode is the ability to target a specific address each time a packet is sent. Unfortunately, API mode requires an API. There are no XBee APIs written for Matlab or AVR-libc, so to avoid hours of coding, AT mode was used with a daisy chain as pictured in **Figure 6**. This allows for global communication without writing several APIs. The downfalls of this setup are 1) Higher latency (up to 7 ms), and 2) Higher chance for communication error for cars further up the chain from the central computer. The latter occurs because there is a slight

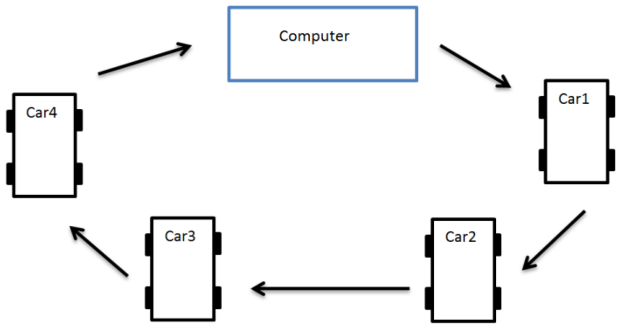


Figure 6: The daisy chain for serial communication via XBees. Commands were sent with the initial byte as the address. If a car reads an address other than its own, it passes the command to the next car.

clock error (.2%) using the AVRs internal RC-oscillator. Latency can be reduced by increasing baud-rate, but this also increases clock error.

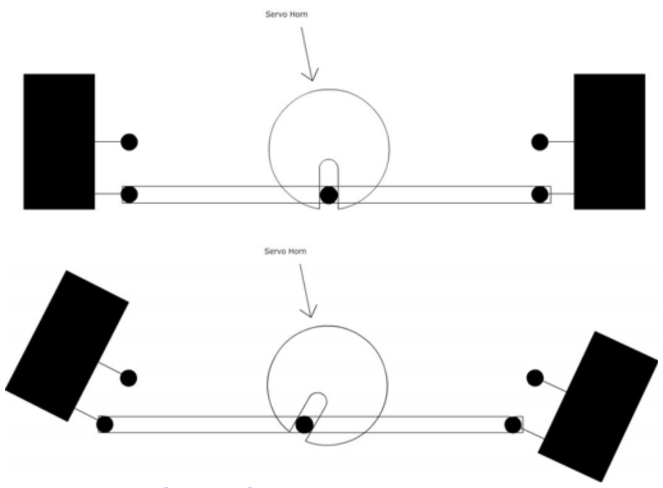


Figure 7: Modification for servo steering control

B. Chassis

The chassis for the cars is an inexpensive R/C car purchased from CVS drugstores. They were $8 each. The cars originally had three state steering and throttle (left/off/right and forward/off/reverse, respectively). The front motor was removed and a servo was mounted to control the steering angle as pictured in **Figure 7**. The body of the car was removed.

C. Controller board

The controller board was designed from scratch using EAGLE. This program was chosen so that the project source could be as accessible as possible. The board is pictured in **Figure 10**. The core logic is an AVR ATmega328. There is a motor driver and isolator chip to remove transient effects on the ATmega328. An XBee wireless module is prominent on the board. Board features include:

- Serial port (UART)
- Wireless (XBee) communication across serial port
- I^2C bus
- Wireless programming
- 2X Servo motor control
- 1X PWM bi-directional DC motor control
- ISP Port for programming
- 8X Analog input pins
- 5X External interrupts

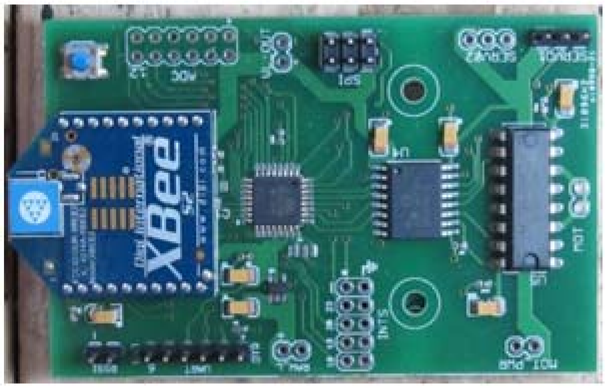


Figure 9: Fully populated custom control board.

The board was printed by Seeed Studios, an "open hardware facilitation company" out of China. The boards were populated in-house.

D. *Vicon Camera System*

A Vicon vision camera system maintained in the Hybrid Dynamics & Control Lab (R. G. Sanfelice, director) at the University of Arizona was utilized to provide location and orientation data streams for each vehicle, updated at a rate of 100 Hz. There are eight MX T20 cameras (**Figure 9**) arranged in a rectangle providing ground coverage of roughly 20' x 22' (**Figure 8**). These cameras operate at a visible red wavelength of 623 nm, receiving reflected energy off of spherical markers attached in a unique spatial arrangement for the four vehicles. The Vicon system Nexus software uses this arrangement to distinguish the separate vehicles regardless of their 6 DOF state in the capture space, and to report the correct data associated with each vehicle. The MX Giganet controller box receives the cables from all eight cameras and performs high speed processing of the vision data before sending the results to the host computer.

Great care was taken to ensure that the four markers on each vehicle were of sufficient uniqueness in arrangement to prevent ambiguous solutions to the multiple vehicle demonstration. This project expanded the HDC knowledge base in multiple vehicle recognition, achieving a stable data stream for four vehicles (previous experience was with two vehicles). It should be noted that on occasion the Vicon Nexus software would experience an ambiguity condition, but only after a vehicle occlusion due to operator intervention while the cameras were running. Each time a simple "pause / live" toggle would fix the issue.

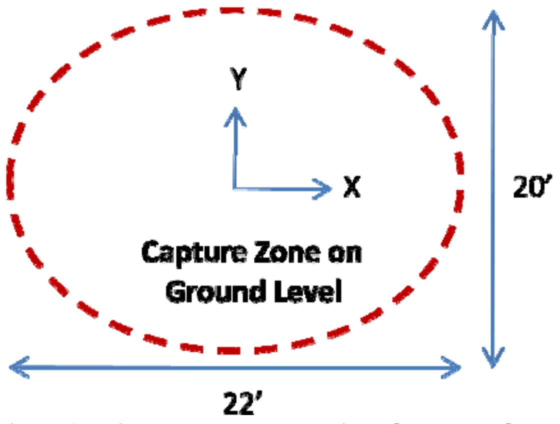


Figure 8: Vicon capture zone projected on ground surface at the HDC lab.

Figure 10: Vicon MX T20 camera with 320 annular visible red LEDs

V. Results and Conclusions

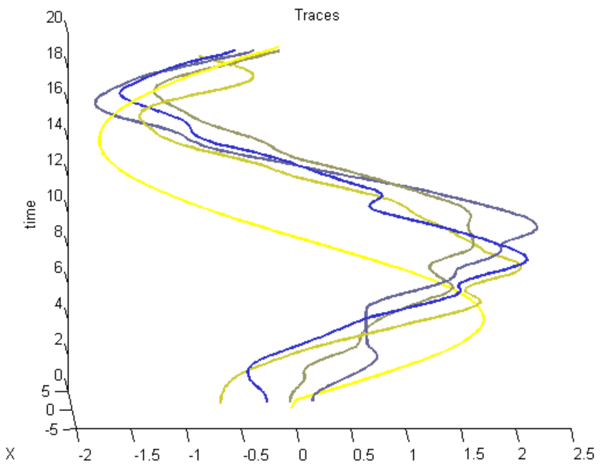


Figure 11: The circular trajectory of four cars following a 5th injected car. The yellow trace is the injected car. Vertical axis is time.

A. *Results*

A plot of the results is pictured in **Figure 11**, where the vertical axis is time. Robots successfully followed a circular trajectory for some initial conditions. These conditions were alluded to earlier—the cars perform best if they are initially oriented in the same direction. Cars have a relatively large turning radius, so they cannot react as well as in the Flash simulation to quickly approaching neighbors. If a collision occurs near head-on the cars will be stuck, as the throttle value never becomes negative.

The flock's trajectory is phase shifted from the injected cars trajectory. This is because the velocity of the injected car is greater than the maximum velocity of the actual cars. Setting the injected cars velocity slightly faster than the maximum cars velocity is advantageous as it maintains a heading that is always in front of the flock, rather than behind it. If the flock heading changes drastically or it is behind the flock there is a higher chance of collisions within the flock as cars reorient to the new heading.

B. Future Development

This project is very expandable. Some ideas for future expansion include the following:

Wireless: An external oscillator of the correct value will limit the clock error on serial communication to very low values (as little as .001%) and at very high baud rates. It would also be advantageous to write a XBee APIs for Matlab and AVR-libc. This would fix latency across all cars to a small value (~1ms for 9600 Baud).

Distribute Processing: It would be very exciting to allow the AVRs to do the computation, with the central computer only providing positional data to each car. With distributed processing, one could limit information to the cars and more easily create an inhomogeneous group.

Additional sensors: The controller boards have 8 analog input channels which one could use to have the cars interact with each other and the environment. The ultimate goal is to have cars locate their neighbors without the use of an external camera system, though this would be very difficult. In addition to analog sensors, a number of external interrupt channels have been broken out which could be used to detect collisions.

General swarm dynamics: The characteristic of the squashing function should be investigated to see how its alteration affects the dynamics. Also, the mapping of the desired vector onto steering and throttle values should be investigated. The mapping is only affected by the current positional data; adding integral and derivative terms may be advantageous.

Predator/obstacle avoidance: We attempted to add a predator to the environment by which the cars would always be repelled. This was done by changing the squashing function for the predator to always be negative by increasing C to a value greater than 1. The cars were in fact repelled, but at the cost of colliding with each other. It may be possible to achieve the desired dynamics by tweaking the squashing further. For instance reducing $(d\sigma/dr)$ near $r = 0$ may reduce collisions by repelling cars less.

C. Conclusions

Given the limitations imposed by the nature of a semester project in a graduate level course, this work is presented as an introduction to the algorithm, simulation, and physical modeling of flocking behavior. As noted in the previous Future Development section, there are many avenues for building on the foundation of the current effort. The XBee wireless controller boards are designed to be flexible and can be used to control a variety of mobile agents, including more sophisticated ones.

One of the key challenges in arriving at the results presented herein is the mapping of the desired direction vector to Ackerman steering input. Other simulations often simply ignore this issue by allowing instantaneous turns and zero radius maneuvers, both of which are not possible with the physical systems. Another somewhat related challenge was keeping the physical mobile agents within the limited Vicon camera system capture zone. The current vehicles use a little less than one meter to perform a full turn, and the HDC lab Vicon capture area is just under 7 meters, giving a ratio of 7:1, which appears to be sufficient to demonstrate stable flocking behavior.

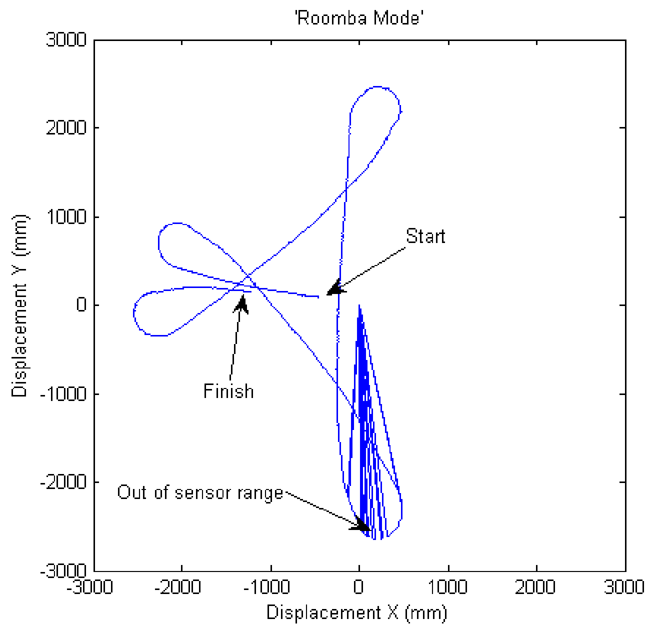
References

[1] H. G. Tanner, A. Jadbabaie, and G. J. Pappas. "Stable Flocking of Mobile Agents, Part I: Fixed Topology," *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, December 2003

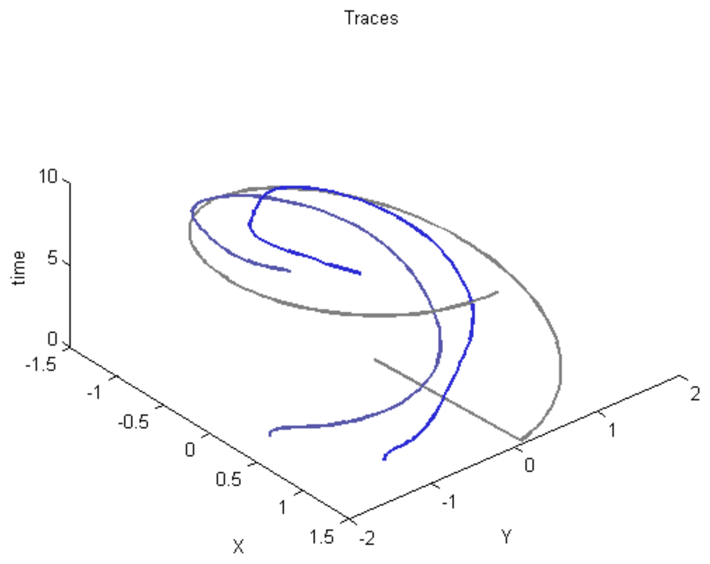
[2] H. G. Tanner, A. Jadbabaie, and G. J. Pappas. "Stable Flocking of Mobile Agents, Part II: Dynamic Topology," *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, December 2003

[3] R. Olfati-Saber, "Flocking for Multi-Agent Dynamic Systems: Algorithms and Theory," *Submitted to the IEEE Transactions on Automatic Control*, June 2004

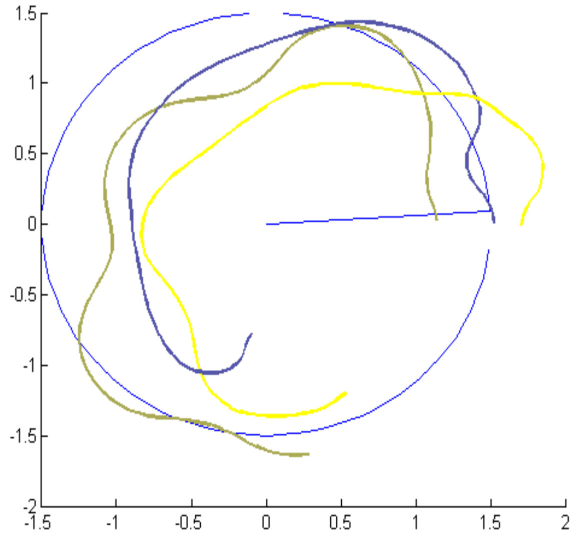
[4] S. Strogatz TED talk on sync, filmed Feb 2004, posted Dec 2008. *(http://www.ted.com/talks/steven_strogatz_on_sync.html)*



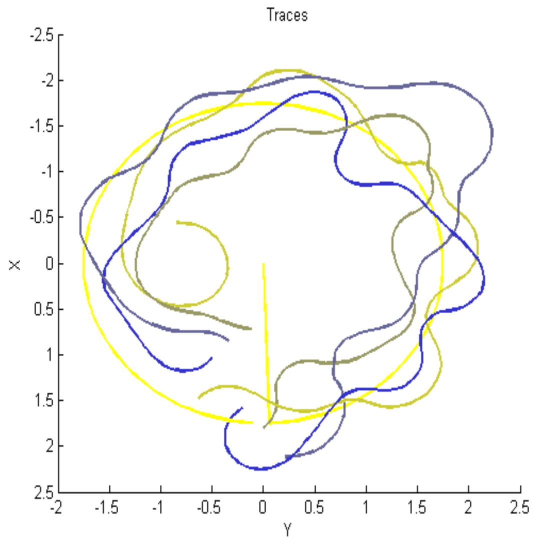
Gallery Figure 1: The first task was to keep a single car within a set raduis.



Gallery Figure 2: Two Cars following circular trajectory.



Gallery Figure 3: Three cars following a circular trajectory.



Gallery Figure 4: Four cars following a circular trajectory