Hierarchical leader-follower network with cyclic pursuit

Sridhar Reddy Velagala[#1], Vedant R. Mahulkar[#2], Dr. Chaouki T. Abdallah[#3]

ECE6563

Georgia Institute of Technology,

Atlanta, Georgia, USA

[1]svelagala7@gatech.edu

[2]vmahulkar3@gatech.edu

[2]ctabdallah@gatech.edu

***Abstract*— This project report presents an approach to simulate a network of robot agents to achieve a behaviour like that of planets and moons revolving around a star. A two-level hierarchy of this behaviour is defined and implemented in the Robotarium to run on robots with unicycle dynamics. The results show reasonable convergence to the desired motion.**

***Keywords*— cyclic pursuit, hierarchical cyclic pursuit, planetary motion, multi agent systems, leader-follower systems**

I. Introduction

The core idea of this project is to simulate the planetary motion of planets and moons with a network of robots by implementing a hierarchical leader-follower behaviour where followers are in a cyclical pursuit around their leader. In essence, the simulation will approximate the planetary motion in a solar system, with moons (followers) circling around a planet (sub-leader) which in turn circle around a star (a central leader).

A. Cyclic Pursuit

In the case of a typical cyclic pursuit, N robot agents connected in the form of a directed cyclic graph (Figure 3) are perpetually in "pursuit" of its neighbour. This results in the robots performing a cyclic motion about their centroid.

B. Hierarchical Cyclic Pursuit

In the case of hierarchical cyclic pursuit, there are multiple levels of cyclic pursuits in the system. Let's denote the levels by L1 (the group formed by the subleaders and the main stationary leader) and L2 (the group formed with the sub leaders and their followers)

For the purpose of this project, the simulation will have one L1 group and three L2 groups.

C. Additional planetary motion dynamics

The interaction of the planetary objects in a system is governed by the gravitational laws, which has been greatly simplified in this project by simply relying on the Euclidean distance (proximity) of the robots to the other robots of interest. This will be used to implement the extension for this project: 1) proximity-based follower switch and 2) comet (stray agent) introduction into the network

II. Main Arguments

A. Robotarium dynamics

Robots (or agents) are considered to be a single point that can freely move in a 2d space. First, the velocity vectors for these point agents are calculated at each time step in order to achieve the hybrid hierarchical cyclic pursuit, then these 2-dimensional velocity vectors (x and y components of velocity) are transformed into unicycle dynamics (v: linear and w: angular, velocities) based on the robot configuration used in the robotarium.

B. Implementation of Simple cyclic pursuit problem, revolving around a circle

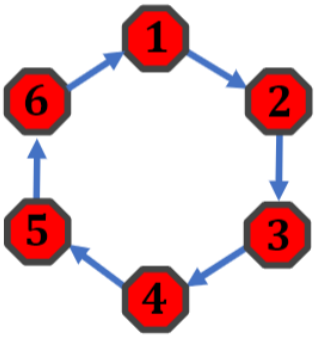
Initially, a simple implementation of a 3-agent cyclic pursuit was developed. The network is a cyclic directed network with agents staying a fixed distance from its neighbour while moving with a velocity tangential to the perceived circular path (Figure 4). Perturbations to this pursuit are corrected by dynamically adjusting the angle of the velocity vector away from the mean (tangent to the perceived circle) based on the distance to its neighbour as shown in Figure 2.

	1	2	3
1	1	0	-1
2	-1	1	0
3	0	-1	1

Figure 1. Graph Laplacian for a 3-agent cyclic pursuit

$$\theta_i = \frac{\pi}{N} + k_r(2r \sin\frac{\pi}{N} - \|x_j - x_i\|)$$

Figure 2. Velocity angle update equation for an N-agent cyclic pursuit



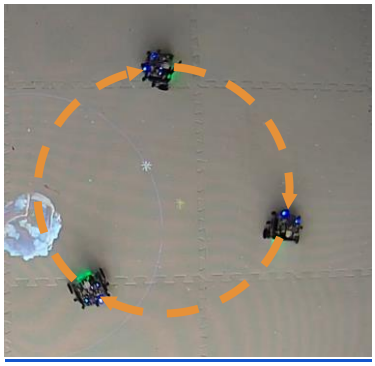


Figure 3. (top) Cyclic graph representation for 6 agents

Figure 4. (right) Cyclic graph representation for 3 robots in the Robotarium

C. Hybrid hierarchical cyclic pursuit - single level

To contrast with the cyclic pursuit described above, we introduce a leader agent (robot) in the centre of the cyclic formation and have the robots circle around the leader while still avoiding collisions with neighbours. To achieve this, additional edges were added from the leader agent to each of the follower agents like a directed star graph (Figure 6). As shown in the Laplacian below, for a particular row, the -1 values denote the connections for the star network (leader to follower) and the 1's represent the connections in the cyclic network (between adjacent followers with the same leader)

	1	2	3	4
1	0	0	0	0
2	-1	0	0	1
3	-1	1	0	0
4	-1	0	1	0

Figure 5. Graph Laplacian (single level 1-leader 3- followers)

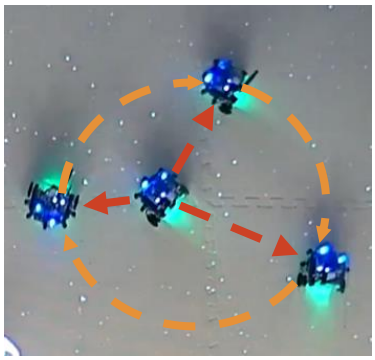


Figure 6. Hierarchical cyclic pursuit with a single leader and 3 followers

In contrast to the generated velocity vector for a simple cyclic pursuit where the velocity is tangential to the perceived cyclic path the agents are intended to follow, we have local leaders to every follower that can take advantage of this new connection for traversing an orbit (circle like path). In this implementation, each agent is influenced by the type of connection of its neighbour (every agent except for the first agent only has two neighbours, one from its leader and one from the follower in the same group).

The first edge (connection) from an agent's leader always moves the robot perpendicular to the line connecting them while staying at a particular distance thereby generating a velocity vector. This angle of this vector w.r.t its leader (90 degree is the normal value) is changed based on its distance to the leader based on a similar formula to equation 1 to account for noise.

The purpose of the second edge from an agent's follower neighbour is to maintain a specified distance (calculated based on the number of follower agents). This case does not apply to a single follower case since there are no other followers in the

subnetwork for possible collisions. These two effects are added and thus dictate the path the robot intends to take.

D. Hybrid hierarchical cyclic pursuit - 2 level

Basing the structure on the single level graph Laplacian, adding more levels to the structure involves generating a block diagonal matrix that encompass each of the smaller Laplacian networks all the way to the highest level. For a 2-level network with 1 leader, 3 sub-leaders, and {1,2,3} follower agents for each of the sub-leaders. Looking at the resultant graph Laplacian (Figure 7), one can assume that each row represents the type of connection for a particular agent: either to their leader (-1) or to their cyclic neighbour in pursuit (1). Here, agent 1 (see row 1) is the central robot with no neighbours and thus is not influenced by any of the other agents and is the highest leader of the network. Then we have agents at positions 2,4, and 7 (-1 values in their respective rows) which have connections from agent 1 and thus form the sub-leader star network. These sub-leader agents (2, 4 and 7) also have connections between themselves (See 1 values in their rows) which form the directed cyclic network. The last level of agents (denoted by the dotted square matrices for each of the {1,2,3} follower agent groups) are connected to their respective sub-leaders (at positions 2, 4 and 7) denoted by (-1) in their rows (star network). The follower agents are also connected amongst themselves (1's within the dotted block matrix) to form the directed cyclic network. Finally, as shown in the single level hierarchical network, the star network aims to rotate the agent at a specified distance from the leader while the cyclic network aims to maintain spacing between follower agents.

	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	0	0	0	0
2	-1	0	0	0	0	0	1	0	0	0
3	0	-1	1	0	0	0	0	0	0	0
4	-1	1	0	0	0	0	0	0	0	0
5	0	0	0	-1	0	1	0	0	0	0
6	0	0	0	-1	1	0	0	0	0	0
7	-1	0	0	1	0	0	0	0	0	0
8	0	0	0	0	0	0	-1	0	0	1
9	0	0	0	0	0	0	-1	1	0	0
10	0	0	0	0	0	0	-1	0	1	0

Figure 7. Graph Laplacian for a 2-level network with 1 leader, 3 sub-leaders, and {1,2,3} follower agents for each of the sub-leaders respectively

E. Implementation of the dynamic node connection modifications based on the leader-follower proximity

After the implementation of the static multi-level cyclic hierarchical pursuit problem, the next step was to implement the provision for the followers to switch their cyclic pursuit group, based on the relative proximity to the sub-leaders. The idea here is to check for each follower node, the relative distance to the sub-leaders. If the follower is sufficiently closer to the sub-leader in question (controlled by a threshold parameter) the follower will switch its cyclic pursuit group to that sub-leader's group. This scheme was chosen to approximate the gravitational interactions between the moons-planets.

Once the proximity detection step (sub leader identification for the followers) is executed, the system Laplacian needs to be modified to indicate these updates. The task is formulated in terms of node k switching its sub leader from node i to node j. The original Laplacian before applying the described transformation is as given below:

	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	0	0	0	0
2	-1	0	0	0	0	0	1	0	0	0
3	0	-1	1	0	0	0	0	0	0	0
4	-1	1	0	0	0	0	0	0	0	0
5	0	0	0	-1	0	1	0	0	0	0
6	0	0	0	-1	1	0	0	0	0	0
7	-1	0	0	1	0	0	0	0	0	0
8	0	0	0	0	0	0	-1	0	0	1
9	0	0	0	0	0	0	-1	1	0	0
10	0	0	0	0	0	0	-1	0	1	0

Figure 8. The Laplacian before the follower switch transformation

To illustrate the algorithm, the following example case has been used in the images supplementing the pseudo code: node 10 switches its leader from node 7 to node 2.

F. Algorithm for switching sub-leader for node k from node i to node j

1. Change the leader identification for node k in Laplacian, i.e.
    a. Set L (k, i) = 0
    b. Set L (k, j) = -1
2. Remove the older traces of node connections for the node k
    a. Replace the 1s in the kth row and kth column with 0
3. Overwrite the new cyclic subgroups (based on updated -1's) with the template cyclic pursuit matrix elements
    a. These subgroups are identified by the presence of -1 terms in the ith column, i.e., the leader's column.

	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	0	0	0	0
2	-1	0	0	0	0	0	1	0	0	0
3	0	-1	1	0	0	0	0	0	0	0
4	-1	1	0	0	0	0	0	0	0	0
5	0	0	0	-1	0	1	0	0	0	0
6	0	0	0	-1	1	0	0	0	0	0
7	-1	0	0	1	0	0	0	0	0	0
8	0	0	0	0	0	0	-1	0	0	1
9	0	0	0	0	0	0	-1	1	0	0
10	0	-1	0	0	0	0	0	0	1	0

Figure 9. Step 1

	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	0	0	0	0
2	-1	0	0	0	0	0	1	0	0	0
3	0	-1	1	0	0	0	0	0	0	0
4	-1	1	0	0	0	0	0	0	0	0
5	0	0	0	-1	0	1	0	0	0	0
6	0	0	0	-1	1	0	0	0	0	0
7	-1	0	0	1	0	0	0	0	0	0
8	0	0	0	0	0	0	-1	0	0	0
9	0	0	0	0	0	0	-1	1	0	0
10	0	-1	0	0	0	0	0	0	0	0

Figure 10. Step 2

	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	0	0	0	0
2	-1	0	0	0	0	0	1	0	0	0
3	0	-1	0	0	0	0	0	0	0	1
4	-1	1	0	0	0	0	0	0	0	0
5	0	0	0	-1	0	1	0	0	0	0
6	0	0	0	-1	1	0	0	0	0	0
7	-1	0	0	1	0	0	0	0	0	0
8	0	0	0	0	0	0	-1	0	1	0
9	0	0	0	0	0	0	-1	1	0	0
10	0	-1	1	0	0	0	0	0	0	0

Figure 11. Step 3

G. Implementation of the comet entry dynamics

This problem builds on top of the follower switch problem, along with the requirement of a node addition algorithm in the existing network connections, based on the proximity between the said node and the sub leaders.

The algorithm will enable the Laplacian to absorb the disconnected "comet" node depending on the proximity with the sub leaders. The comet will start the simulation as a disconnected node, and after some time, will be sent flying towards the planetary system. If the comet comes in "proximity" of any of the sub leaders, it'll get absorbed in the cyclic pursuit group of that leader.

This task is formulated in terms of addition of node k to leader i's sub-group. The original system Laplacian before applying this transformation is given below:

	1	2	3	4	5	6	7	8	9	10	11
1	0	0	0	0	0	0	0	0	0	0	0
2	-1	0	0	0	0	0	1	0	0	0	0
3	0	-1	1	0	0	0	0	0	0	0	0
4	-1	1	0	0	0	0	0	0	0	0	0
5	0	0	0	-1	0	1	0	0	0	0	0
6	0	0	0	-1	1	0	0	0	0	0	0
7	-1	0	0	1	0	0	0	0	0	0	0
8	0	0	0	0	0	0	-1	0	0	1	0
9	0	0	0	0	0	0	-1	1	0	0	0
10	0	0	0	0	0	0	-1	0	1	0	0
11	0	0	0	0	0	0	0	0	0	0	0

Figure 12. The system Laplacian before applying node-addition transformation

To illustrate the algorithm, the following example case has been used in the images supplementing the pseudo code: node 11 gets added to the cyclic pursuit subgroup of node 2.

H. Algorithm for adding node k to sub-leader node i

1. Change the leader identification for node k in Laplacian, i.e.
    a. Set L (k, i) = -1
2. (Same as step 3. from the follower switch algorithm) Overwrite the new cyclic subgroups (based on updated -1's) with the template cyclic pursuit matrix elements
    a. These subgroups are identified by the presence of -1 terms in the ith column, i.e., the leader's column.

	1	2	3	4	5	6	7	8	9	10	11
1	0	0	0	0	0	0	0	0	0	0	0
2	-1	0	0	0	0	0	1	0	0	0	0
3	0	-1	1	0	0	0	0	0	0	0	0
4	-1	1	0	0	0	0	0	0	0	0	0
5	0	0	0	-1	0	1	0	0	0	0	0
6	0	0	0	-1	1	0	0	0	0	0	0
7	-1	0	0	1	0	0	0	0	0	0	0
8	0	0	0	0	0	0	-1	0	0	1	0
9	0	0	0	0	0	0	-1	1	0	0	0
10	0	0	0	0	0	0	-1	0	1	0	0
11	0	-1	0	0	0	0	0	0	0	0	0

Figure 13. Step 1

	1	2	3	4	5	6	7	8	9	10	11
1	0	0	0	0	0	0	0	0	0	0	0
2	-1	0	0	0	0	0	1	0	0	0	0
3	0	-1	0	0	0	0	0	0	0	0	1
4	-1	1	0	0	0	0	0	0	0	0	0
5	0	0	0	-1	0	1	0	0	0	0	0
6	0	0	0	-1	1	0	0	0	0	0	0
7	-1	0	0	1	0	0	0	0	0	0	0
8	0	0	0	0	0	0	-1	0	0	1	0
9	0	0	0	0	0	0	-1	1	0	0	0
10	0	0	0	0	0	0	-1	0	1	0	0
11	0	-1	1	0	0	0	0	0	0	0	0

Figure 14. Step 2

I. Post-processing of the velocity signal

The velocity values generated by the network are first thresholded to a maximum value. Then they transformed into unicycle dynamic outputs to match the robots in the robotarium. Finally, control barrier functions are applied to these robots to actively avoid collisions with other robots, work area boundaries, and potential obstacles. This is then sent to execute on the robot.

III. Results and Conclusion

Referenced videos show various behaviours of the network. These include stable hierarchical motion, follower swap, and comet entry [5,6,7]. Visually, the agents tend to follow the expected path and become gradually more stable after the initial transition from being on a circle to following the leader. However, the constraints posed by the control barrier functions sometimes lead to situations where a follower robot gets trapped with other groups of robots. The follower robots also influence the velocity of the leader robots especially when they are within the constraints of the barrier functions. These issues can be solved by using a larger radius for the robots and fewer robots overall. But it is to be noted that the constraint on the test area dimension limits the maximum radius of various levels. Another issue is when the follower robots frequently pass each other instead of staying apart in a circular formation. This could be tuned by manually varying the weights on how much each of the effects (separation distance between followers vs speed of orthogonal motion with respect to the leader) should be. It should also be noted that these parameters vary with the number of followers, cycle radius, and speed.

Another thing to note is that the current follower switch condition sometimes leads to excessive follower-sub leader switches, and it can be prevented by fine tuning the threshold parameters for the switch condition accordingly. This tuning will depend on the radius of the cyclic pursuit levels along with the barrier certificate boundary conditions.

While a fair bit of experimentation and evaluation was carried out on the scheme developed in the project, a lot of scope for improvement and addition of new features exists. The comet entry simulation performed in this project assumes that the comet will get attached to the smaller cyclic pursuit groups that are revolving around the sub leaders. But the scheme can be updated to include the mass information of all the objects and with appropriate usage of mass as weights the comet can either join the smaller levels of the cyclic pursuit around the sub leaders or it could join the larger cyclic pursuit around the central robot, Moreover, a lot of experimentation and tests can be carried out on the system to analyse the system stability and repeatability.

Acknowledgment

We would like to extend our sincere thanks to Prof. Chouki T. Abdallah and Prof. Sean Wilson, for their guidance throughout the project duration, right from the selection of the topic for the project to the completion of the project. A special thanks is also due for the Robotarium facility being made accessible to us, which was pivotal for the purpose of our project simulation and experimentation.

References

[1] Smith, Stephen L., et al. “A Hierarchical Cyclic Pursuit Scheme for Vehicle Networks.” Automatica, vol. 41, no. 6, 2005, pp. 1045–1053., https://doi.org/10.1016/j.automatica.2005.01.001.

[2] Sean Wilson, et al., "The Robotarium: Globally Impactful Opportunities, Challenges, and Lessons Learned in Remote-Access, Distributed Control of Multirobot Systems," in IEEE Control Systems Magazine, vol. 40, no. 1, pp. 26-44, Feb. 2020.

[3] “ECE6563 - Georgia Institute of Technology.” School of Electrical and Computer Engineering at the Georgia Institute of Technology, https://www.ece.gatech.edu/courses/course_outline/ECE6563.

[4] M, Vedant, and Sridhar Reddy V. “Sridhar701Pitt/hcyclicpursuit_robotarium.” GitHub, 2021, https://github.com/Sridhar701Pitt/HCyclicPursuit_robotarium.

[5] Hierarchical Leader-Follower Network with Cyclic Pursuit, YouTube, 11 Dec. 2021, https://www.youtube.com/watch?v=V3A8EueX8dg. Accessed 11 Dec. 2021.

[6] Hierarchical Cyclic Pursuit Test v0.3g, YouTube, 11 Dec. 2021, https://www.youtube.com/watch?v=CgonLMsBqc0. Accessed 11 Dec. 2021.

[7] CyclicHierarchy CometConvergence v0.1, YouTube, 11 Dec. 2021, https://www.youtube.com/watch?v=bKfrZWEaq-E. Accessed 11 Dec. 2021.

Appendices

Algorithm 1: Hybrid Hierarchical Cyclic Pursuit

```
velocity = 0;
for currentRobot = 2 to N do
    neighbours = topological_neighbours(laplacian, currentRobot)
    for neighbour to neighbours do
        if neighbour is leader then
            velocity = velocity + move perpendicular to leader
        else
            velocity = velocity + stay x distance from neighbour
        end
    end
end
velocity = transform_to_unicycle_dynamics(velocity)
velocity = maximum_velocity_limits(velocity)
velocity = barrier_function_constraint(velocity)
send_to_robot(velocity)
```

Figure A. Pseudocode for the Hybrid Hierarchical Cyclic Pursuit using Leader Follow