Scouting and Defense in Capture the Flag using Distrubuted Controls with Multi-agent Systems

Edgardo Jose Marchand
ECE 6563
Georgia Institute of Technology
Atlanta, Georgia, USA
emarchand3@gatech.edu

Eren Yildirir
ECE 6563
Georgia Institute of Technology
Atlanta, Georgia, USA
eyildirir3@gatech.edu

Tianyang Cao
ECE6563
Georgia Institute of Technology
Atlanta, Georgia, USA
tcao77@gatech.edu

***Abstract*—This paper discusses the design of distributed controls for multi-agent systems to accomplish two tasks, exploration and defense of objectives, in the game of capture the flag. The documents introduce leader-following dynamics, formation control, and cycling pursuit algorithm for multi-agent systems without the use of central controls and with information limited to relative distances between neighboring agents. For exploration, a modified version of formation control with leader following allows for groups of agents to move around an arena collecting information. For defense, a variation of consensus with orientation offset and a combination of leader-following, formation control and consensus dynamics allows for efficient defensive tactics.**

Keywords—multi-agent system, distributed control, network control, formation control, consensus equation, weighted dynamics, leader following, rotational offsets.

I. Introduction

Capture the flag is a game consisting of two teams trying to be the first to find, capture, and bring back the opponent team's flag to their base. The game can be divided into multiple tasks or stages: protect the flag, explore the environment, collect information, create decoys to break through opponent's defense, fetch the opponent's flag, and setup communication network. These tasks need to be accomplished for a team to successfully win a match. A form of the game has been utilized to study robotic and network algorithms. The game has been found to be useful to experiment with distributed controls with many agents on a team because the teams could contain hundreds of participating agents. The number of agents involved makes it impractical for utilizing a central control and most agent do not need to make extensive calculations. This paper will address two tasks, exploration and defense, using leader, consensus, and formation control algorithms with distributed controls.

II. Game Tasks

A. Exploration

Exploration in Capture the Flag consists of sub-groups of a team whose main task is to explore the environment in the arena and relaying that information to the team. The recon team needs to move around the arena discretely recording data on the environment in the arena like obstacles and paths as well as search for the location of the opponent's flag. They can also distribute information about the movements of the opposing team. Since exploration teams are prioritizing information they should not engage or get trapped by the opponent's team. Exploration is a key facet of the games that allows the team to construct efficient strategies to challenge the opposing team.

B. Defense

The defending sub-group allows for proper protection of a team's flag. This group must protect their flag from any attempt by the opponent to steal the flag. To do so, the defending squad must react to incoming attacks and prevent any member of the opponent team from reaching the flag. In the event that the flag is snatched the defending sub-group must prevent the opponent from running away with it. For defense there are multiple strategies that could prevent an opponent from getting to the flag. One of them is to form a defense formation that will not allow the enemy team from reaching the flag but allows for friendlies to move around. The second one is to react to incoming enemies and somehow disable them. These strategies can be implemented simultaneously which will allow for not only protection but also lowering the number of active enemies.

III. Assumptions

Before developing distributed control algorithms, multiple assumptions must be presented to allow the algorithm to function properly. Specific assumptions have been made in relation to the robots used, the amount of information available for the robots, the environments and game logistics.

A. Types of Robots

The robots used for this paper are tricycle robots with maximum speed of 0.5 m/s. Their dimensions of the robots are 0.1 x -.1 meters and are modeled as unicycles with the mapping between models done internally. There will be two classes of robots defined as leaders and followers. Leaders possesses higher capabilities of communication and processing as they decide the actions of a group of followers. Leaders are equipped with enough memory to store information such as location of opponent's flag and any obstacle encountered. They also store the information gathered from its group. Leaders are equipped with sensors that measure distances between objects, recognition capabilities that allow it to distinguish between enemies, friendlies, and obstacles, and memory to store

locations of obstacles, enemies, and flags or waypoints. Leaders also have a disk field of vision of radius 0.6 meters around them but do know the distances between itself and its sub-group's followers without line of sight. Followers on the other hand, are low memory and limited communication agents where they only transfer information to the leader but do not receive information back. They do not store information once it has been sent to the leader. As for hardware, follower class robots contain measuring sensor to detect distances between itself and objects. They have the capabilities to distinguish between friendly agents, enemy agents, obstacles and flags. They have a disk field of view with radius of 0.6 meters around them to detect objects. They do however have access to the distances between itself and its sub-group's members without line of sight. Figure 1 depicts the range and size of the robots used for this experiment.

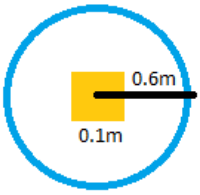

Figure 1. Robot size and range of sensor assumptions

The data available to leader agents are relative distance between itself and other agents and location of flags, enemies (if encountered), and obstacles. They also possess a greater communication network with other sub-group's leaders. In the other hand, followers only have access to the distances between its own sub-group's agents and itself as well as any objects in its field of view. Robots do not have access to their own position nor the actual position of other agents, only the relative distance between itself and an object. Also, they do not possess a compass to tell their own orientation.

B. Enviornment

There was no significant environment assumption made as the control algorithm can succeed in almost any type of environment. The main assumption is that the algorithm will run in a planar arena and no air-borne units are allowed. The is also the assumption that no obstacle will be placed directly next (touching distance) to either of the flags. This is to ensure that a circular motion around the flag is possible. The overall arena could have any number of obstacles as long as there exists open paths for the robots to pass through. The arena is to be 2 meters by 3 meters long and the boundaries count as obstacles. Figure 2 depicts the environment in which experiment ran with relation to robot size.

Figure 2. Arena size in relation to robots for experiment.

C. Relation to Game

For the overall scope of the game, the assumption has been made that multiple tasks can be performed at any given time. This would mean exploration and defense can run simultaneously assuming they do not share agents. Agents should not change roles in the middle of a running task but can be recycled into different roles after their task has been completed. The exploration task is to be used early on if not the first thing a team should do at the beginning of a match. Meanwhile, the defense task should always be running. All rules of the game apply to these tasks as they are strategies to be implemented during a match.

IV. Controllers

For each, scouting and defending, a set of controllers were designed for both follower robots and leader robots to accomplish the appropriate task. Scouting uses formation control and leader following dynamics, while defending uses cyclic pursuit with formation control, leader following, and consensus dynamics.

A. Scouting

For scouting the playing field, a three-agent scouting party that can hold formation while roaming a set of predetermined waypoints was implemented. This strategy was chosen because it allows for minimal computation for follower dynamics and allows for large area coverage. This strategy also allows for more than one group of scouting agents to be active at any given time, minimizing the amount of time taken for arena exploration. The set of waypoints followed by the leader is to be set in advance as a plan for exploration. It does not matter what the actual waypoints are since the robots are able to adapt to any environment and only need to reach "close" to the actual waypoint. Only having a leader dictate and hold relevant information reduces the computational power of the overall group as the follower agents only need to observer the surrounding, send any information to the leader, and follow the movements of the leader.

The two followers were set to hold a specific distance from their two neighbors, forming a rigid formation. This allows for the formation to hold no matter the initial conditions, as well as, maintain the formation in the presence of obstacles. The leader was set to head for the next waypoint with a constant speed, but also slow down if it got too far from its neighbors.

Both simulations and Robotarium experiments verified the correct behavior of this controller design. Figure 3 shows a visual representation of the formation held by the agents.

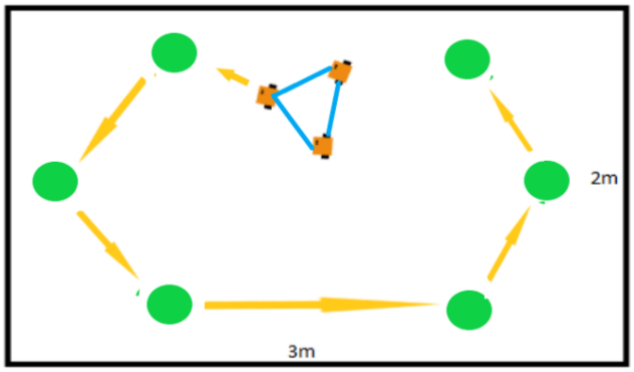

Figure 3. Formation control and leader-following dynamics following pre-defined waypoints used for exploration. The green spots correspond to waypoints while yellow arrows correspond to the likely path the leader will take.

The follower algorithm for holding the formation was, as follows:

$$\dot{x}_i = K_f \sum_{j \in N_i} \| (x_j - x_i)^2 - d_{ij}^2 \| (x_j - x_i)$$

Where $(x_j - x_i)$ is the distance neighbor j is from agent i, d_{ij} is the offset defined as the set distance the robots should be apart, K_f is the formation gain that was later optimized for the robots in the Robotarium. Note that K_f is determined by the physical constraints of the system, meaning it would be different for another kind of robot. The idea is to perform consensus but separate the agents by a set distance. These distances were defined prior to running the dynamics and where chosen to set a formation. Since the leader has different dynamics, the follower dynamics will never reach equilibrium but will hold the formation with respect to the movements of the leader.

The leader followed a similar algorithm that combined formation holding and waypoint movement, as follows:

$$\dot{x}_l = w_f \sum_{j \in N_l} \| (x_j - x_l)^2 - d_{lj}^2 \| (x_j - x_l) + w_w \frac{(x_{\text{wpt}} - x_l)}{\| x_{\text{wpt}} - x_l \|}$$

Where x_{wpt} is the location of the waypoint, w_f and w_w are the corresponding formation and path weights. They were later optimized for then robots in the Robotaium. w_f was found to be K_f /2. w_w is determined by the physical constraints of the system since it scales a unit vector and would be different for another kind of robot. The leader dynamics still contains the formation control based on consensus with distance offset but also incorporates consensus with a pre-defined location. Since this second term is weighted more heavily, the leader will prioritize going to the waypoints but making sure the formation with the followers remains constant.

For this to work, a complete graph must be implemented. The complete graph guarantees the formation will hold no matter the number of agents involved. It also prevents the graph from becoming disconnected in the presence of obstacles. Another important aspect of this control design is that it is scalable. You can have any number of agents as long as its above 3 follow this implementation successfully. Adding more agents to the group allows for greater area of coverage as they move through the arena. This implementation also prevents the leader from leaving his followers behind and exposing him. Lastly, the control design is easily adaptable and if needed could add an extra term to the leader dynamics to make him move away from an enemy if an enemy is spotted. The communication network allows for the leader to hold all information found by the scouting group and, per assumptions, can transfer that to other groups in the team.

B. Defense

Two strategies were implemented for defense of the flag. First, robots are constantly moving in a circular motion around their own flag as a form of protection. Second, there is a 3-robot fixed-formation patrol team that always tries to block the direct path between the flag and any incoming opponent. This cycling pursuit tactic is utilized because it eliminates any static gaps for intruders to use to reach the flag. The rotation around the flag also allows for constant area coverage near the flag. Meanwhile, the small radius of the rotation does not allow for any standard size enemy robot to pass through. While the cycling pursuit algorithm protects the flag by circling around it, the tringle formation allows for reactive protection method by intercepting enemy agents at the half point between the enemy and the flag. This allows the group to react to different attacks and will always position themselves to intercept the enemy. If an enemy continues to move toward the flag, the triangle formation could, if desired, merge with the cycle pursuit creating an even tougher barrier for the enemy. If possible, the triangle formation could try to disable the enemy if the enemy gets a certain distance from the flag. Both of these strategies could be scaled as the number of agents does not affect the performance as long as at least three agents are in a group. In theory, it is possible to have multiple intercept squads roaming around the area of the flag.

In order to run the defense strategies, two controller algorithms that achieve cyclic pursuit, formation control, and consensus were implemented. The first controller is designed to achieve the cyclic pursuit movement with agents as shown in Figure 4.

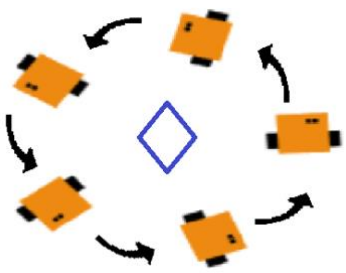

Figure 4. Cyclic pursuit algorithm with counter clockwise movement

The following node level dynamics show that all the robots will move around a fixed-radius circle if they are placed evenly around in a circle initially:

$$\dot{x}_i = R(\theta)(x_{i+1} - x_i),\ i = 1, \ldots, N-1,$$

$$\dot{x}_N = R(\theta)(x_1 - x_N),$$

Where i is the current robot's index, i+1 corresponds to the immediate neighbor of the robot i, N is the total number of robots, and R(θ) is a rotation matrix defined as

$$R(\theta) = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

In order to achieve fixed-radius cyclic pursuit, θ is

$$\theta = \frac{\pi}{N}$$

for all robots for all times. Ignoring the rotation matrix, the dynamics of the agents is simple. The agents try to reach the position of their neighbor but since they are all connected in a line, the robots endlessly follow each other. To get the circular motion for each robot, the dynamics is multiplied by a rotation matrix. If we multiply out the term $R(\theta)(x_{i+1} - x_i)$ and do some algebraic manipulations, we can see that the $\dot{x}_i$ vector has the direction tangential to the circle at each time instant for each robot only if θ = π/N. Such tangential velocity direction of each robot gives rise to a perfect cyclic movement. If θ > π/N, then the velocity direction of each robot will point more outward than the tangential line, and thus robots will move outward away from the center of circle. On the other hand, if θ < π/N, then the velocity direction will point more inward than the tangential line and create an inward movement towards the center of circle. However, after testing this controller design, the actual model of the robots would move around a circle with gradually increasing radius due to small offsets. In order to fix this issue, we modified our controller design to:

$$\dot{x}_i = R(\theta_i)(x_{i+1} - x_i),\ i = 1, \ldots, N-1$$

$$\dot{x}_N = R(\theta_N)(x_1 - x_N)$$

$$R(q_i) = \begin{bmatrix} \cos q_i & \sin q_i \\ -\sin q_i & \cos q_i \end{bmatrix}$$

$$q_i = \frac{p}{N} + err$$

$$err = r_{desired} - r_{actual-i}$$

where, $r_{desired}$ is the desired radius of circle to be maintained, and r_{actual} is the actual distance between the current robot and the center of circle (flag location). The only difference with the previous design is that now the θ value for each robot at each time step changes dynamically in order to correct the error in maintaining the desired circle radius. A positive err term (caused by a robot being closer to the flag than desired) would make θi larger than π/N, which in turn drives the robot outward away from the flag, and vice versa for a negative err term. Matlab simulations and Robotarium experiments verified the correct behavior of this controller design.

The second controller is designed to achieve the desired behavior of the 3-robot patrol team. Same as in the scouting task, the three robots consist of one leader and two followers. The followers' sore task is to follow the leader while maintaining the formation. Therefore, their controller dynamics is exactly the same as that in the scouting section (The desired formation is the same as well). The differences come from the controller design for the single leader robot. Here, the leader needs to constantly block the direct path between our own flag and one moving opponent robot while maintaining formation with the followers. Figure 5 illustrates the proper behavior.

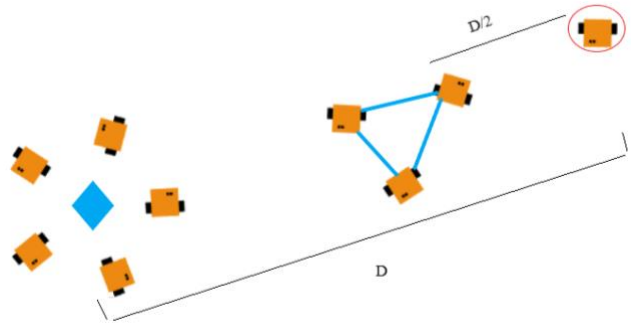

Figure 5. Illustration showing 3-robot squad intercepting in between flag and enemy with D being the distance the enemy is to the flag. Red circle indicates an enemy, while blue means friendly objective.

To achieve this, we designed a controller for the leader that combines formation control with a consensus dynamics, as follows:

$$\dot{x}_l = W_f \cdot \sum_{j \in N_l} \left\| (x_j - x_l)^2 - d_{lj}^2 \right\| (x_j - x_l) + W_c \cdot \left[(x_{flag} - x_l) + (x_{oppo} - x_l) \right]$$

where x_l is the leader position, d_{lj} is the desired inter-agent distance for the formation, x_{flag} is the flag position, x_{oppo} is the moving opponent robot's position, and W_f,W_c are the weights of the two terms. The first term of this controller is the formation control term discussed in the scouting task. The second term is the consensus dynamics. It drives the leader to the middle point between the flag and the opponent. One way of illustrating it is to rewrite the consensus term as:

$$(x_{flag} - x_l) + (x_{oppo} - x_l) = 2 \times (\frac{x_{flag} + x_{oppo}}{2} - x_l) = 2 \times (x_{mp} - x_l)$$

where x_{mp} is the position of the middle point between flag and opponent. This directly shows that the consensus term provides a corrective control effort to drive the leader to the middle point. Finally, combining the formation control term with the consensus term achieves the desired behavior of the leader. Moreover, there's a weight for each term that determines the relative importance of formation maintenance versus blockade/consensus. In our design, we want the leader to focus more on blocking the enemy (consensus) than maintaining formation with followers since the task here is defense. We set W_f to 3.75 and W_c to 2. Although W_f seems to be larger than W_c, it actually doesn't mean formation control is more important than consensus in this case. Here, since we have very small sized formation comparing to the distance between flag, leader, and opponent, the absolute magnitude of the consensus term is much larger than the formation control term. Therefore, a larger W_f is used to compensate for this inequality in absolute magnitudes. If the absolute magnitudes are scaled to the same level, then W_c would be much larger than W_f, and thus giving more importance to enemy blockade over formation maintenance. Both simulations and Robotarium experiments verified the correct behavior of the patrol team controller designs.

CONCLUSION

This paper has presented multiple distributed control implementation to achieve certain tasks in the game of capture the flags. Using formation control and leader following dynamics, a successful controller implementation for scouting areas was achieved. The idea was to use formation control to form a set formation that allows for maximum area coverage while one robot dictates the movement of the formation through the arena. The use of consensus dynamics with offset allows for formation building, as long as, the offsets are chosen carefully to achieve proposed formation. The leader, meanwhile, uses combination of formation control and consensus with static points to move the formation from waypoint to waypoint without breaking the constraints of the formation. On the other hand, using consensus dynamics with rotation offset allows for cycling pursuit to be performed a form of protection of the flag. The implementation consisted of having consensus run with only one neighbor to each agent forming a loop, and a carefully constructed rotation matrix to create and endless circular motions that agents perform around the flag. This was supplemented with the scouting algorithm but with changes made in leader dynamics to track incoming enemies. Instead of using waypoint, the leader uses the distance to enemy agents and with the location of the friendly flag known, the leader would intercept between the enemy and the friendly flag a distance half of the distance between the flag and the agent. These two implementations are easily adaptable and scalable as they can be implemented with any number of agents, as long as its greater than three. They also do not use a central control and are achieved with minimal computation and information.

ACKNOWLEDGEMENTS AND REFERENCES

Special mention should be given to Professor Pierpaoli from Georgia Institute of Technology, who teaches Network Controls. His lectures are the basis of this document and explores the use of different distributed controls.

[1] P. Pierpaoli, 'ECE 6563: Network Controls', Georiga Institute of Technology, Atlanta, Georiga, U.S, 2018