Capture the Flag: Distributed Control System Design

Jorge Alberto Rodriguez
ECE Grad Student
alberto.rod@gatech.edu

Rabeya Jamshad
ECE Grad Student
rjamshad3@gatech.edu

Alfredo Garó
ECE Grad Student
garomonegro@gatech.edu

Al Chandeck
ECE Grad Student
a.chandeck@gatech.edu

***Abstract*—This project, implements a two team capture the flag game. The consensus algorithm is used to implement three multi-agent robotics behaviours including formation control, cyclic pursuit, and single agent exploration in a local and distributed manner.**

***Index Terms*—Distributed Control, Network Control, Swarm Robotics**

I. Introduction

Multi-Agent Robotics is an area of wide research and application ranging from terrain exploration and mapping to humanitarian de-mining. In order for agents to successfully complete the required task in a robust manner, it is imperative that the system design is local and distributed. The consensus equation makes use of the agents immediate neighbors, as described by a network graph, and the relative distance between the agent and its neighbors, to cause all agents to converge at a point. The equation can be modified through the design of weights and energies to achieve unique behaviors by the robots.

The game "Capture the Flag" (CTF) is implemented using multi-agent dynamics. The game's main objective is to get the enemy's flag while at the same time protecting yours. Our version of the game is won when one team either successfully captures the flag of the other team or defends its own. Our team dynamics are based on formation control to defend the flags of both teams A and B, exploration by a single node of Team B to find the flag of Team A and cyclic pursuit by Team A to defend its flag from the attacking robot of Team B. Control Barrier Certificates, which ensure collision prevention, are used to ward off the attack by the agent from Team B. Therefore, the goal of Team A is to keep the Team B agent from reaching Team A's flag, whereas, the goal for Team B is to protect its flag, find the location of and attack Team As flag.

II. Assumptions

A. The Field

The field in which the game will be played is assumed to be parallel to the the plane formed by the x and y axis. This implies that the dynamics of the robots will be designed in a two dimensional world. The field measures roughly 3.2 by 2.0 meters, theoretically the boundaries of the field do not affect the dynamics rather just the time it will take the robots to reach their destination. It is typical to see closed bases and obstacles along the field in CTF, but for this version of the game it is assumed that the field and the bases are open spaces. Although, obstacles may be added as the Control Barrier Certificates would take care of that as well.

B. The Robots

The experiment was realized with the aid of the Robotarium robots, which are modeled as kinematic unicycles. However, it is simpler to describe the dynamics of robots by simulating them as single integrators. Therefore, the dynamics and controllers for the robots are designed as single integrators. Then a transformation is used to convert control outputs for single integrator robots to the equivalent control inputs for the kinematic unicycles available at the Robotarium.
From the setup of the Robotarium lab, the exact position of all robots can be accessed at any given time. However, we made the following assumptions to simulate local and distributed algorithms:

- The robots can sense the enemy agents that are within a certain range, Δ. This means that, robots can only measure the relative distance of other robots within a radius Δ, thus mimicking a Δ-disk graph. Thereby, the robot j that will be neighbor of robot i satisfies the equation:
$$j \in Ni \text{ iff } \parallel Xi - Xj \parallel \leq \Delta$$
Therefore, N_i is the set of neighbors of agent i.
- Robots on the same team have a communication channel through which they can communicate with every other robot in the team. This is used to switch agent dynamics between different behaviours when enemy's agents are detected

III. Methodology

This game (CTF) can be segregated into 4 stages: Protection, Exploration, Decoy and Fetch. Figure 1 shows the different states that represent the stages implemented in our project (Protection and Exploration\Decoy) and the switching conditions between different behaviours.

For the protection of the flag, formation control is the basic approach in the realm of multi-agent robotics. This behavior by itself is inefficient since we would need a large number of agents to fully complete the task and cover the complete perimeter of the base. A more effective strategy will therefore, be to use periodic motion of the agents to construct a barrier

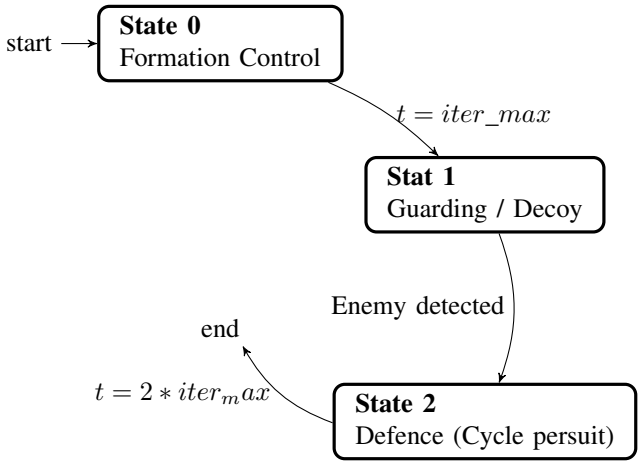


Fig. 1: Machine state of the behaviors implemented

against enemy invasion. Cyclic pursuit is implemented to achieve this goal.

Without previous knowledge of the environment, exploration is needed to discover the location of the enemy's flag. This requires implementing efficient exploration strategies that will allow a team to discover the enemy flag without having to arbitrarily sweep the entire environment.

IV. Simulation

A. Formation Control

Formation control is used to form an initial defensive position around the flags of both teams. Formation control of the $N = 5$, agents in each team alone however, does not guarantee that the required formation will assemble at the desired location and with the desired orientation. To ensure that, two phantom nodes are added to the network graph of each team. These phantom nodes are stationary anchor nodes and through edge weights between phantom nodes and agents, orientation and localization of the formation of both teams is guaranteed. The first phantom node is placed at the center of the formation which is also the location of the flag of each team. The second phantom node orients the formation to face the opposing team for best attack and defense formation. This means that the network topology is defined through the interconnection between seven ($n = N + 2$) nodes, of which two nodes are phantom nodes and do not contribute to the behaviors of the team beyond ensuring correct formation.The formation for each team is represented in Fig 2. Here, node 6 and 7 are the phantom nodes, used to center the and orient the formation respectively.

The minimally infinitesimally rigid (MIR) condition for formation requires that we have at least $2n - 3$ edges. For a MIR between $n = 7$ nodes , $2n - 3 = 11$ edges are needed. However, a MIR does not ensure unique formation from all initial conditions. Therefore, a complete graph K_7, as shown in Fig 3, network topology is used. The Laplacian and Weight

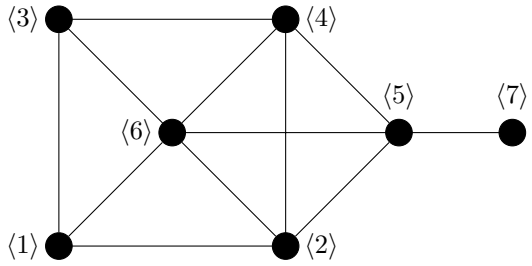


Fig. 2: Formation Geometry

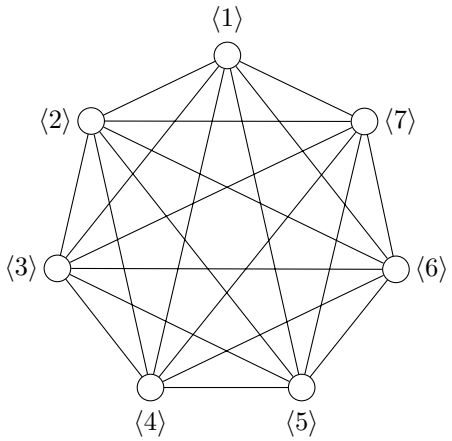


Fig. 3: Complete Graph

Matrices for Formation Control are given as:

$$L = \begin{bmatrix} 6 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & 6 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & 6 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 6 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & 6 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & 6 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & 6 \end{bmatrix}$$

$$\mathrm{W} = \begin{bmatrix} 0 & 0.5 & 0.5 & 0.707 & 0.743 & 0.353 & 0.838 \\ 0.5 & 0 & 0.707 & 0.5 & 0.320 & 0.353 & 0.390 \\ 0.5 & 0.707 & 0 & 0.5 & 0.743 & 0.353 & 0.838 \\ 0.707 & 0.5 & 0.5 & 0 & 0.320 & 0.353 & 0.390 \\ 0.743 & 0.320 & 0.743 & 0.320 & 0 & 0.450 & 0.1 \\ 0.353 & 0.353 & 0.353 & 0.353 & 0.45 & 0 & 0.55 \\ 0.838 & 0.390 & 0.838 & 0.390 & 0.10 & 0.55 & 0 \end{bmatrix}$$

The formation dynamics are referenced in Algorithm 1. Here, $w_{ij} \in W$ are the required distances of the edges $e_k \in E_f$ such that the formation graph $G_f = (V, E_f, W)$ completely defines the formation.

Variables and constants:
N: number of agents in each team
$N_i(L)$: topological neighbors of agent i
dX_i: velocity vector of agent i

Algorithm 1 Formation Control Algorithm

procedure FORMATION(L, W)
  for $i \leftarrow 1, N$ **do**
    $dX_i \leftarrow \begin{bmatrix} 0 \\ 0 \end{bmatrix}$
    for $j \in N_i(L)$ **do**
      $dX_i \leftarrow \sum_{j \in N_i} \left(\| X_j - X_i \|^2 - w_{ij}^2 \right) (X_j - X_i)$
    end for
  end for
end procedure

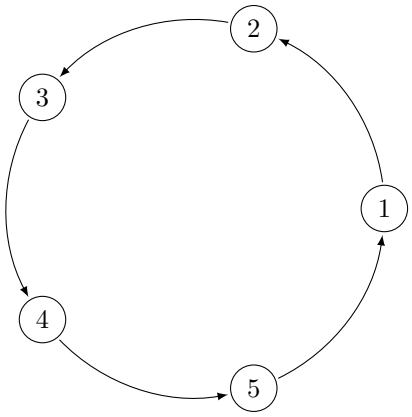


Fig. 4: Circle Graph

B. Cyclic Pursuit

In order to efficiently protect the flag with a small number of robots, we incorporated cyclic pursuit dynamics (CPD). Algorithms for common implementations of cyclic pursuit are derived from the dynamic equations that describe them. The equations 1 and 2 fully describe the CPD.

$$\dot{x}_i = -\begin{bmatrix} \cos(\frac{\pi}{n}) & \sin(\frac{\pi}{n}) \\ -\sin(\frac{\pi}{n}) & \cos(\frac{\pi}{n}) \end{bmatrix} (x_i - x_{i+1}) \tag{1}$$

$$\forall \ i = 1, ..., N-1$$

$$\dot{x}_N = -\begin{bmatrix} \cos(\frac{\pi}{n}) & \sin(\frac{\pi}{n}) \\ -\sin(\frac{\pi}{n}) & \cos(\frac{\pi}{n}) \end{bmatrix} (x_N - x_1) \tag{2}$$

Along with a change in dynamics for this behavior, the communication topology of the robots is also changed from the Complete graph (Fig. 3) to a directed Circular graph (Fig. 4). These result in the robots revolving around a circle.

However, we have further modified the simple CPD in order to properly defend a team's flag. The pseudocode of our implementation is shown in Algorithm 2.
Notice in Alogrithm 2 we have the constants Kp_1, Kp_2, and D which are not present in simple CPD algorithm implementations. The latter variable represents the nominal distance that consecutive agents in the cycle graph must have in order for all to lie evenly distributed across the circumference of the circle they describe. This distance depends on the radius,*r* of the desired circle and the number of agents, *N* as shown below:

$$D = 2r * sin(\pi/N)$$

The purpose of the constant Kp_1 is to ensure that the robots revolve with a constant radius. Kp_1 plays the role of a proportional factor that multiplies the error between the nominal inter-agent distance, *interdist*, and the actual interagent distance. On the other hand, the constant Kp_2 ensures that agents resist perturbations from other attacking agents that push them away when they get closer due to the barrier functions and ensures the robots revolve around the same center at all times.

The constants Kp_1 and Kp_2 require trial and error tuning. For our experiment, with *N* = 5, we found that Kp_1 = 6 and Kp_2 = 0.9 work just fine.

Algorithm 2 Cyclic Pursuit algorithm

procedure CYCLIC(L) ▷ Laplacian L of Cycle digraph
  for $i \leftarrow 1, N$ **do**
    $dX_i \leftarrow \begin{bmatrix} 0 \\ 0 \end{bmatrix}$
    for $j \in N_i(L)$ **do**
      $\alpha \leftarrow \pi/N + Kp_1(D - \| X_j - X_i \|)$
      $R \leftarrow \begin{bmatrix} cos(\alpha) & sin(\alpha) \\ -sin(\alpha) & cos(\alpha) \end{bmatrix}$
      $dX_i \leftarrow dX_i + R(X_j - X_i) - Kp_2(X_i - C)$
    end for
  end for
end procedure

Variables and constants:
N: number of agents in the cycle graph
$N_i(L)$: topological neighbors of agent i
α: heading direction of agent i
Kp_1: gain for interagent angle
D: normal interdistance between neighbors
R: rotation matrix
dX_i: velocity vector of agent i
Kp_2: center-preserving controller gain
C: location of the center of the circle (flag location)

C. Decoy

One agent from each team was selected to perform the exploration and decoy stages of the game. Figure 5 shows the control flow of the dynamics of the selected agent. After the formation control stage is done, the selected agent moves away from its own flag in order to discover the enemies. Once the enemies are discovered, the selected agent will change its dynamics in order to move in direction of the centroid of the set of enemies detected. This comes from the assumption that the other team will be guarding their flag. Finally, if the attacking agent is close enough to the enemies' flag, it will aim to reach the flag position evading any possible enemy in its way.

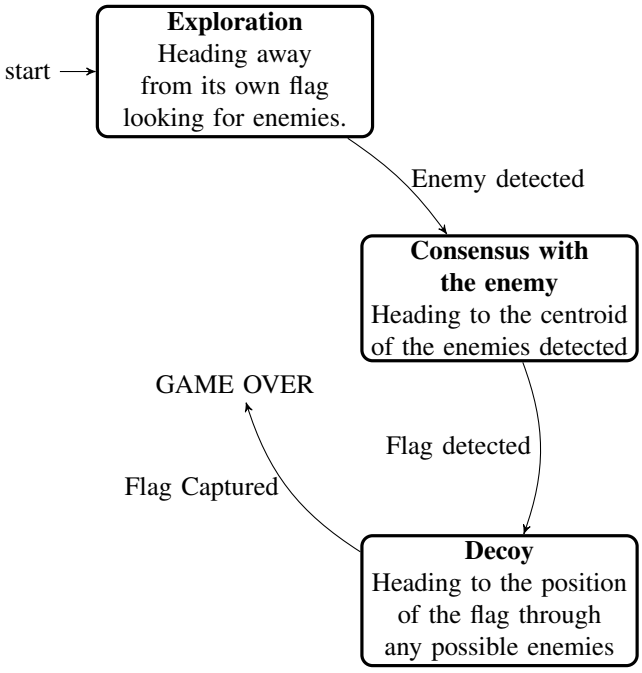


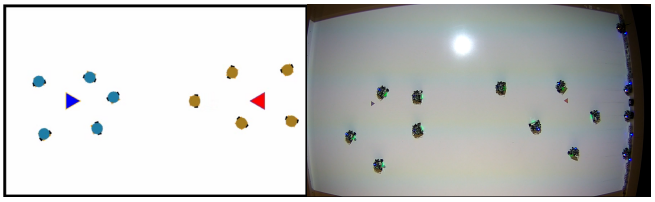
Fig. 5: Exploration and decoy diagram

Algorithm 3 shows our implementation of this stage of the game. It is important to mention that all dynamics mentioned in this stage are local. As the attacking robot just knows the relative position of the enemies and flag that are within its sense radius.

Algorithm 3 Decoy algorithm

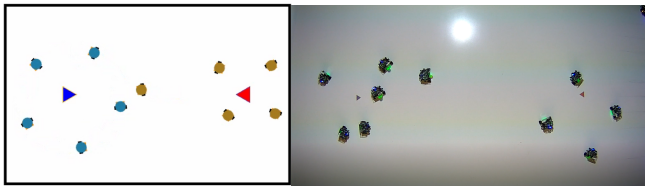
procedure DECOY( )
  $dX_A \leftarrow 0.04(x_A - C)/ \parallel x_A - C \parallel$
  if $\parallel x_A - C \parallel > \Delta$ **then**
    if $\mid N_{en} \mid > 1$ **then**
      $dX_A \leftarrow \begin{bmatrix} 0 \\ 0 \end{bmatrix}$
    end if
    for $j \in N_{en}$ **do**
      $dX_A \leftarrow dX_A + 0.3(x_j - x_A)$
    end for
  else if $\parallel x_A - C \parallel < \delta$ **then**
    Print "Flag Captured"
  else
    $dX_A \leftarrow 0.5(C - x_A)$

Variables and constants:
dX_A: velocity vector of agent A
X_A: position vector of agent A
N_{en}: set of enemies within sensing range from A
Δ: A's sensing range
C: location of A's own flag
δ: arbitrary small distance error



(a) Simulation (b) Robotarium

Fig. 6: Formation Control Results



(a) Simulation (b) Robotarium

Fig. 7: Exploration and Cyclic Pursuit Results

V. RESULTS

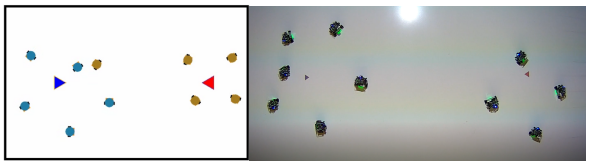
The algorithm was run on the robotarium simulator in MATLAB and the behaviours required for the game verified. The algorithm was then run on the Robotarium Robots.The simulation begins with agents from both teams randomly distributed across the robotarium. The results of the three algorithms: formation, cyclic pursuit and exploration and decoy are discussed below.

A. Formation Results

The simulation results shown in Fig. 6 display that both teams correctly center themselves around their respective flags, orienting themselves to intersect the invading enemy. However, we see that in the robotarium experiment despite reaching the correct formation the two teams have still not corrected their orientation and center. This is because of the delay in the robots finding all their team members when starting from arbitrary initial conditions in the robotarium. Due to the four minute limitation on the robotarium runtime, the formation control in our algorithm was not allowed an arbitrary long time to correct its orientation.

B. Exploration and Cyclic Pursuit Results

Once the formation is made, a single agent from one of the team begins exploration for the the enemy nodes by moving away from its own flag. When the explorer node comes within the sensing ratio of the defending nodes, the defending nodes use its communication channel to begin circular pursuit to ward off the enemy using control barrier certificates. This behaviours is observed in (Fig. 7). Once again, a lag is observed between the simulation and the robotarium results. However, agents of the same team can also be observed pushing away the invading node using control barrier certificates as they form a circular defence around the flag.



(a) Simulation (b) Robotarium

Fig. 8: Decoy and Cyclic Pursuit

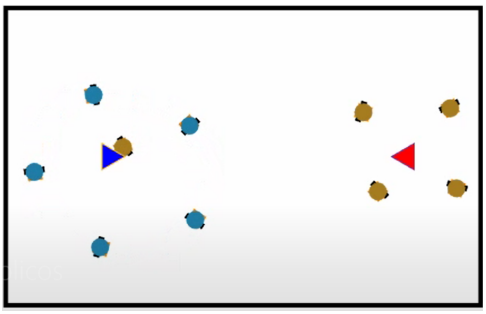


Fig. 9: Flag Capture

C. Decoy

The circular pursuit of the defending robots however, leaves an inter-agent open space that is not guarded by the barrier certificates. As a result, the invading enemy node can mask itself as part of the circular formation and find its way to the flag. Fig. 8 shows the invading node breaking through to the circular pursuit which is now completely formed by the defending nodes and Fig. 9 shows it successfully capturing the flag.

VI. Conclusion

This project therefore, utilizes the local and distributed properties of control algorithms and control barrier certificates to implement three different behaviours. The project explored localization and orientation of multi-agent formation by extending the network graph beyond physical nodes. Circular pursuit for unicycle robots was implemented using the required and actual inter-neighbour distance and finally an effective exploration and invasion algorithm was implemented to allow the enemy node to locate and capture the opponent's flag successfully. A possible lag in the implementation of behaviors between simulation results and robotarium results was also observed leading to discrepancies between the two.