

# RoboFlag – A Framework for Exploring Control, Planning, and Human Interface Issues Related to Coordinating Multiple Robots in a Realtime Dynamic Environment

**Atif I. Chaudhry**

Mechanical & Aerospace  
Engineering  
Cornell University  
139 Upson Hall  
[aic7@cornell.edu](mailto:aic7@cornell.edu)

**Prof. Raffaello D’Andrea**

Mechanical & Aerospace  
Engineering  
Cornell University  
101 Rhodes Hall  
[rd28@cornell.edu](mailto:rd28@cornell.edu)

**Prof. Mark Campbell**

Mechanical & Aerospace  
Engineering  
Cornell University  
208 Upson Hall  
[mc288@cornell.edu](mailto:mc288@cornell.edu)

## Abstract

*RoboFlag is a game and associated hardware and software framework under development at Cornell University. This paper gives an introduction to the game and an overview of the hardware and software necessary for the framework. A general discussion follows of various research topics that have already been addressed with RoboFlag. The main goal of this paper is to demonstrate that RoboFlag can be used to investigate the fundamental issues necessary for successful coordinated control of multiple robots in a realtime dynamic environment.*

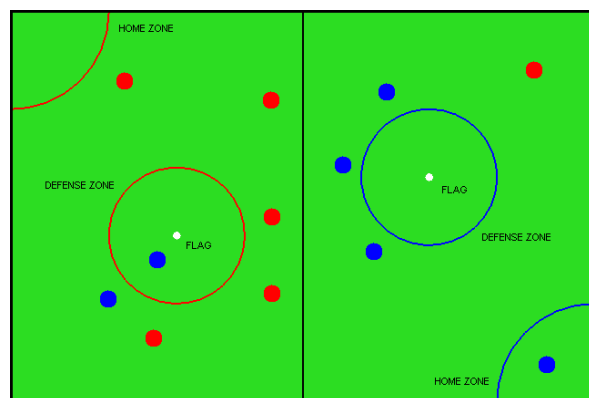
## 1. Introduction

Cornell University has successfully fielded teams in the International RoboCup competition for several years. Building on this experience, a new game has been formulated at Cornell called RoboFlag. This new game uses a larger, more specialized, field than RoboCup and has a more complex scoring structure which in turn may lead to more complex strategies. In support of this new game the necessary hardware and software has been developed. In addition, a software simulation of the game was created and is undergoing continued revision and upgrading. The game and framework can serve as a starting point for many different areas of research regarding the issues of control, planning, and human interface arising from the coordination of multiple robots in a realtime dynamic environment.

## 2. RoboFlag Game Overview

RoboFlag is a game loosely based on “Capture the Flag” and “Paintball.” Two teams play the game, the Red Team and the Blue Team, and are refereed by the Arbiter. Each team’s objective is to infiltrate the other team’s territory, grab the other team’s flag, and bring it back to their own Home Zone. The game is thus a mix of offense and defense: secure the opponent’s flag, while at the same time prevent the opponent from securing your flag. The field is depicted in figure 1.

Points may be scored in several ways. The largest payoff occurs when an opponent’s flag is safely brought back to the Home Zone. Points may also be scored by “tagging” an opponent in designated areas of the playing field.



**Figure 1: RoboFlag Field**

## 2.1 Playing Field

The playing field consists of a carpeted area of specific length and width. The field is divided in half with each half having three distinct zones: the Home Zone, the Defense Zone, and the Attack Zone. The Home Zone is a quarter of a circle located at one corner of the half field. This is a safe haven for whichever team's territory corresponds to this half of the field. The Defense Zone is a circle that the corresponding team is trying to defend. The team's flag sits in the center of this circle and the team can not enter the Defense Zone. The remainder of the half field is the Attack Zone and is where the team can attack/tag the opposing team when it enters to try to capture the flag.

## 2.2 Game Rules

The following refers to the Blue Team's robots, similar definitions hold for the Red Team's robots. Each robot is in one of four game states: active, flagged, tagged, or inactive. These states, and the transitions between states, are described below. The state transitions are also represented in Figure 2.

**Active:** This is the normal operating state for a robot and is the default state when the game begins. When active, a robot can tag other opponents for points and capture the opponent's flag. A Blue robot is set to the active state if

1. It is in the Blue Home Zone.

**Flagged:** The robot is in this state when it has captured the opponent's flag. Only one robot per team can be in this state. The difference between the active and the flagged states is that a flagged robot can be tagged in more areas of the field. A Blue robot becomes flagged if

2. No other Blue Robots are flagged, it is active, and the distance between it and the center of the Red Defense Zone is less than radius of the flag.

**Tagged:** A Blue robot becomes tagged in four ways:

3. When it is active, in the Red Attack Zone or the Red Home Zone, and an active Red Robot comes within tagging distance of it.
4. When it is flagged, anywhere on the playing field with the exception of the Blue Home Zone and the Blue Defense Zone, and an active Red Robot comes within tagging distance of it.
5. When it is flagged anywhere in the Red Half and a flagged Red Robot comes within tagging distance

of it.

6. When it is active or flagged, and it comes within tagging distance of any robot that has been inactive or tagged for longer than the grace period.

A Robot in the tagged state cannot be controlled, communicated to, or send information to the other robots. When tagged, the arbiter assumes control of the robot and guides it back to the robot's Home Zone.

**Inactive:** The Robot is in this state when it has violated one of the rules of the game. A Blue Robot is immediately deemed inactive if it is active or flagged, and any of the following conditions are satisfied:

7. If it enters the Blue Defense Zone.
8. If it has traveled too far and run out of fuel since it was last in the Blue Home Zone.

An inactive robot cannot be controlled, cannot be communicated to, and cannot send information to the other robots for the remainder of the half. It remains stationary for the remainder of the half. The Arbiter will immediately take control of the robot in question when inactive. When case 7 occurs and the Robot is inside the Blue Defense Zone, the Arbiter will move the Robot to the closest point on the boundary of the Blue Defense Zone.

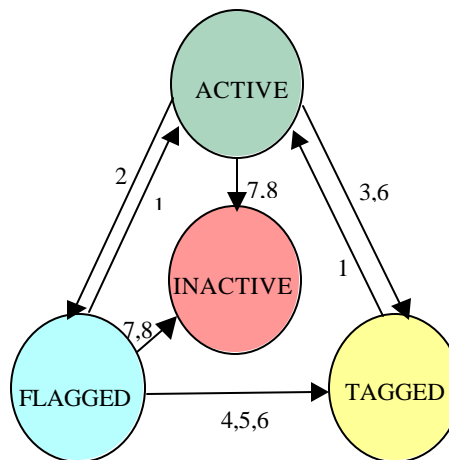


Figure 2: Robot State Transition

## 2.3 Scoring Points

Points are assigned when transition between robot states occur:

**Tagged by an opponent:** One point is assigned to the

Blue Team when a Red robot transitions to the tagged state via transitions 3, 4, or 5, as described in the previous section.

**Tagged by an Arbiter controlled Robot:** Ten points are assigned to the Blue Team when a Red robot transitions to the tagged state via transition 6.

**Capturing the flag:** Five points are assigned to the Blue team when a Blue Robot transitions from the active state to the flagged state.

**Bringing the flag home:** Twenty-five points are assigned to the Blue Team when a Blue Robot transitions from the flagged state to the active state.

**Inactive Robots:** Ten points are assigned to Blue for every inactive Red robot.

The framework allows the point values for each type of transition to be changed. This may be done to promote different strategies. For instance, greatly increasing the points for tagging will lead to strategies that emphasize intercepting attacking enemies over trying to capture the flag.

## 2.4 Information and Control Architecture

Each team is composed of 9 Entities: 8 robot Entities and 1 Central Control Entity. All communication and control information for each team passes through the Arbiter.

Each robot entity receives local information from the Arbiter: its own position, orientation, translational and angular velocity, and game state; and the same information of each nearby objects. Each robot entity sends local information through the Arbiter such as desired robot velocity. All command information and all incoming local information is subject to latency. The Central Control Entity neither sends nor receives local information. It is through the Central Control Entity that a human can interface with the robots to see what they see and to control them to varying degrees. This is covered in more detail in section 4. Each entity can send and receive information to and from any other entity via a communications network.

### 2.4.1 Local Information

Each robot entity receives its own local information, such as position, orientation, translational and angular velocity, and the state of the robot. In addition they receive nearby local information which includes the above information for any objects within their field of view and

for which a direct line of sight exists.

## 2.5 Game Variations

There are several variations on this base RoboFlag game. The first is the number of robots per team. Depending on the size of the robots relative to the field the number may be increased or decreased to avoid crowding or to make the game action simpler. Another variation involves the use of neutral obstacles that are scattered on the field. These obstacles, like robots, have the ability to tag either team's robots. These obstacles can be either stationary or moving in a random walk fashion. A third variation being explored at Cornell involves the use of tagging balls. A robot can tag an enemy from a distance by hitting them with one of these tagging balls. The RoboFlag game and framework have been made amenable to changes to allow for flexibility in using the system to investigate different research areas. For example, research into coordination of swarms would require large number of robots, where as path planning research may only require a few.

Another possible variation is in the capabilities of the individual robots. Normally robots are homogeneous, with the same speed, maneuverability, and sensing ability. The framework can be altered to allow different robots to have different capabilities. Thus there could be a slow but far seeing "sensing robot" and a fast but short sighted "attack" robot. In this case, the information provided by the Arbiter to each robot could also include the robot type of any enemy robot detected. The ability to vary individual robot capabilities allows RoboFlag to be used to investigate a large array of control and planning algorithms.

## 3 RoboFlag Hardware

The development of the RoboFlag hardware benefited from the previous work on RoboCup. The robots currently used for RoboFlag are the same as those developed for previous RoboCup competitions. The RoboFlag field is significantly larger than the regulation RoboCup field. This necessitated the use of a two camera vision system. Each camera sits over half the field and feeds its view of the field into a vision computer. This computer then uses colored dots located on top of the robots to identify each robot, the robot's location, which team it is on, and its

orientation. The vision computer feeds this information to the Arbiter computer which is also connected to the computers running the various robot entity software. The Arbiter gets the commands from these computers and relays them to the actual robots via a wireless RF link. This setup is depicted in figure 3.

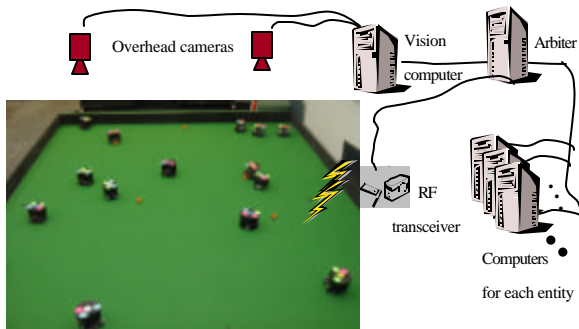


Figure 3: Cornell RoboFlag Hardware Setup

## 4 Human Interface

As stated previously, each team is composed of several robots and the Central Control Entity. It is through this entity that a human operator can see what information is available to his team, and to direct the individual robots to varying degree. Each robot shares its local information. The Central Control Entity pools this information and can display it for the human operator via a Graphical User Interface, GUI. A good GUI shows each robot's location, status, and fuel level. In addition it should allow the human to easily direct the individual robots. Figure 4 shows one example GUI developed at Cornell for RoboFlag.

### 4.1 Robot Location and Status

The GUI shown in figure 4 is primarily composed of a depiction of the playing field. On this field the location of each Blue robot is depicted with a blue icon, either a circle, square, or triangle, corresponding to three different robot types. This information is known since each Blue robot communicates its local information to the Central Control Entity. The status of each robot is conveyed in two ways, by the type of blue icon shown on the field and also at the bottom of the GUI next to each robots name. An active

Blue robot is depicted with a solid blue icon. A flagged Blue robot has a white dot in the center of its blue icon. A tagged Blue robot has a red dot in the center, while an inactive blue robot has a grey dot. Also at the bottom of the GUI, next to a robot's name and state, is a bar showing how much fuel the robot has. As the bar gets shorter it turns from blue to yellow to red to indicate normal, dangerous, and critically low, fuel situations.

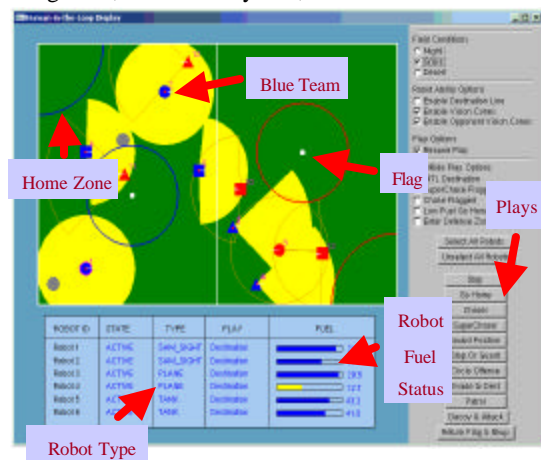


Figure 4: Sample GUI for Blue Team

### 4.2 Vision Information

The vision field of each robot is depicted by a yellow area around the robot. A yellow circle corresponds to the robot having 360 degree vision. For robots with limited vision, the yellow circle is replaced with a yellow wedge shape. An enemy robot is shown as a solid red icon unless it is flagged (white center dot), tagged (blue center dot), or inactive (grey center dot). Based on what type the enemy robot is, their predicted field of view is drawn with a brown outline. This allows the user to see what the enemy sees through their robots' vision. One use of this capability is using your own far seeing robot to shadow or track, undetected, a shorter seeing enemy robot.

### 4.3 Robot Control

The Central Control Entity can be used to control the individual robots. In the GUI depicted in figure 4 this control can be accomplished in two ways. The first is direct control where the operator selects a robot and tells it specifically where to go. To select a robot the operator simply left clicks with the mouse on the circle representing the robot. He then moves the mouse to that robot's desired

destination and right clicks. This sends a destination command to the computer running that robots entity software which uses the destination to guide the robot. The robot uses its own guidance and avoidance algorithms to avoid running into an inactive or tagged robot, or enter its own Defense Zone which would result in becoming inactive.

The second way to control a robot is by using one of several built in offense or defense plays including patrol, circle offense, chaser, and Guard Position. These plays are listed along the right side of the GUI shown in figure 4. Much like a football team, the robots are told what play to run and are then left to implement them on their own.

These approaches are specific to this Central Control Entity and this GUI. The framework can be used to investigate other control algorithms such as biologically inspired or genetic. A user simply has to code these algorithms for the robot entity software. They can be invoked through a new GUI of the users creation, or by taking the default GUI supplied with the RoboFlag simulation software and altering it, perhaps by just adding a button that when pressed, activates the new control code.

## 5 Research Uses

The RoboFlag framework can be used to investigate several different fields related to multiple robot control and coordination.

### 5.1 Path Planning

An operator can guide a robot in several ways. However it is unrealistic for the operator to continually guide each robot, especially as the number of robots increases. In the RoboFlag frame work an individual Robot entity is controlled by giving it a target destination. It is then up to the Robot entity software to plan an appropriate path and execute it. One method tested on Cornell's RoboFlag hardware involves stream functions. The stream functions are used to compose local-extrema free potential fields. This is done by assigning a potential well at the goal destination and a doublet aligned with the flow at any obstacles. These fields are then used to find an appropriate path. [1]

Another possible path planning algorithm involves building a probability map of enemy robot locations based

on the limited sensor data. This can be done by combining known apriori information on the distribution and probability functions related to the enemy robots as described in [2]. There are plans to try such an algorithm on the RoboFlag test bed.

### 5.2 Human Factors

With a human operator controlling several robots the human factor issues become critical in determining how efficiently the team accomplishes its goal. On-going studies are being conducted with the RoboFlag framework to address these issues. One area of investigation is how vehicle speed affects the operator's ability to control the team and how important automation becomes. Numerous games have been run with robot speed set to different values. Game scores and operator feedback and comments are used to gauge the affect speed has on operator performance.

Another issue is whether two human operators allow for more efficient control since they can split responsibility for offence and defense. By having two Central Control Entities per team, the RoboFlag framework was used to play games with two operators controlling each team. The results of these and other Human Factors experiments are given in [3].

### 5.3 Coding Architecture

The RoboFlag framework has been used to investigate various coding architectures. The coding architecture influences the type of control algorithms that can be used for the robots. One type of architecture is called subsumption. With subsumption very simple base behaviors are described. More complex robot behavior is then created by grouping and connecting the base behaviors. This allows for building several complex behaviors. However the interdependency of behaviors made modify a complex behavior difficult. In addition developing new algorithms becomes dependant on older ones.

Another coding architecture demonstrated with RoboFlag is the finite state machine. With a finite state machine a robot acts in a particular way corresponding to a state. The robot behavior changes as the robot changes from state to state under user defined conditions. This allows great flexibility in creating different robot behaviors and in controlling when a robot exhibits a

specific behavior. By adopting a finite state machine architecture, different control algorithms can be written and easily swapped into and out of the framework. They can also be merged with other control algorithms by creating a hybrid finite state machine. This is the current coding architecture used in the Cornell RoboFlag framework. However, others exist and anyone can download the framework and change the coding architecture to suit their own research objectives.

## 6 Conclusions

The field of multiple vehicle coordination is beginning to show great promise. There are many fundamental issues that need to be better understood if multiple vehicle coordination is to achieve its full potential. These issues include control algorithms, path planning, human factors, and software architecture. By building on years of previous work for the international RoboCup competition, Cornell University has devised a new game that highlights the challenges of multiple vehicle coordination. Having a robust RoboFlag framework consisting of the hardware and software to either control real robots, or to simulate them, many of these issues can be further investigated by different research groups with varying focus and expertise. Please see <http://roboflag.mae.cornell.edu> for more information including the complete simulation and framework software.

## Acknowledgment

The research and testbed for this program was supported through the DARPA MICA program (Contract #F30602-01-2-0577), with Lt. Col. Sharon Heise, PhD, as the Program Monitor and Mr. Carl DeFranco from AFRL Rome Lab as the Contract Monitor. Additional support was provided by AFOSR MURI contract #F49620-01-1-0361

The following people have made RoboFlag possible:

### Faculty

[Prof. Raffaello D'Andrea](#), [Prof. Mark Campbell](#), Prof. Richard Murray (Caltech), and Prof. Raja Parasuraman (Catholic University).

### Staff

Dr. JinWoo Lee, Dr. MyungSoo Jun, and Andrey Klochko.

### Graduate Students

Atif Chaudhry, David Schneider, Jeff Sullivan, Matt Earl Steve Waydo (Caltech), Jesse Veverka, and Zain Cheng.

### Industrial Partners

Chris Miller (SIFT), Harry Funk (SIFT), and Robert Goldman (SIFT).

### Past Students/Staff

Michael Babish, Dr. Tamas Kalmar-Nagy, Dr. Julie Adams, Dr. Adam Hayes, Prabhu Ram Raghunathan, Justin Wick, Joran Siu, Nirav Shah, Lyle Chamberlain, and Japeck Tang.

## References

- [1] Stephen Waydo and R.M. Murray, "Vehicle Motion Planning Using Stream Functions." Accepted: 2003 IEEE International Conference on Robotics and Automation, Taipei, Taiwan, May 2003.
- [2] Myungsoo Jun and Raffaello D'Andrea, "Probability Map Building of Uncertain Dynamic Environments with Indistinguishable Obstacles", Submitted to the American Control Conference, Denver, CO, 2003.
- [3] Jesse Veverka and Mark Campbell, "Experimental Study of Information Load on Operators in Semi-Autonomous Systems," submitted to the 2003 AIAA Guidance, Navigation and Control Conference, Austin TX, August 2003.