A Simulation Test Bed for Coordination of Unmanned Rotorcraft and Ground Vehicles

Murat Yasar∗ Derek O. Bridges† Goutham Mallapragada∗ Joseph F. Horn‡
The Pennsylvania State University, University Park, PA, 16802

This paper presents the development of a simulation test bed for a command, control, computer, communication, intelligence, surveillance and reconnaissance (C4ISR) system. The test bed features coordination of a rotorcraft unmanned aerial vehicle (RUAV) and multiple unmanned ground vehicles (UGVs) by integrating a high-fidelity rotorcraft model with actual ground-based robots that emulate UGVs. The RUAV component of the test bed is realized on a networked computer simulation, whereas networked robotics hardware is employed for the UGV representation. The RUAV dynamics run a high-fidelity nonlinear simulation model of a UH-60A Black Hawk helicopter. To emulate the dynamics of the UGVs, several networked Segway robots are employed. The proposed C4ISR system undertakes the task of mission management from the top level discrete-event supervision (DES) to the bottom level continuous-time regulation.

Nomenclature

\( h \) Rotorcraft altitude above ground, ft
\( K_D \) Derivative control factor
\( K_I \) Integral control factor
\( K_P \) Proportional control factor
\( r_b \) Rotorcraft yaw rate, rad/s
\( V \) Rotorcraft total airspeed, kt
\( V_D \) Rotorcraft vertical speed, ft/s
\( \gamma \) Simulated sensor visibility angle, rad
\( \theta \) Angle between center of sensor and vertical axis, rad
\( \phi \) Rotorcraft roll angle, rad
\( \theta \) Rotorcraft pitch angle, rad
\( \psi \) Rotorcraft heading angle, rad
\( v \) Robot linear velocity, m/s
\( \omega \) Robot angular velocity, rad/s

Subscript

\( cmd \) Commanded value
\( err \) Error
\( r \) Rotorcraft
\( t \) Target

I. Introduction

Simulation technologies have become an integral part in the development of aerospace vehicles, systems and operations. Therefore, it is crucial to establish standards for simulation and visualization instruments, data transfer and communication protocols and high levels of simulation fidelity. Substantial research is in

∗Graduate Research Assistant, Department of Mechanical and Nuclear Engineering, 127 Reber Building.
†Graduate Research Assistant, Department of Aerospace Engineering, 233 Hammond Building, Student Member, AIAA.
‡Associate Professor, Department of Aerospace Engineering, 233 Hammond Building, Senior Member, AIAA.

American Institute of Aeronautics and Astronautics
progress for all phases of simulation technologies to support the decision-making process of a pilot or operator. Furthermore, as the aerospace industry continues its research into unmanned vehicles, new problems are posed which modeling and simulation is best suited to answer. A continued effort is being made to reduce the cost of high-fidelity simulation through the use of low-cost components such as personal computers and readily available visualization software such as open source FlightGear.\(^1\)

Future autonomous unmanned air vehicles (UAV) will need to work in teams with other UAVs and unmanned ground vehicles (UGVs) to share information and coordinate activities in much the same way as current manned systems. For this purpose, mathematical models and simulation studies have been developed to understand and ultimately provide this future UAV capability at the Complex Systems Simulation Lab of The Pennsylvania State University.

The research focuses on coordination and cooperation for autonomous UAVs and UGVs engaged in information gathering and data fusion tasks, including cooperative tracking, area exploration, and target search. The underlying mathematical model for coordination and cooperation employs quantitative measure of regular languages\(^2\) and discrete event supervisory (DES) control.\(^3,4\) This language-theoretic approach builds on established principles of sensor data fusion for event generation, extending the ideas to problems for control of heterogeneous agents. The research has made substantial progress towards formulating, solving, and demonstrating these methods for UAV-UGV coordinated operation. In particular, distributed algorithms that enable UAV based surveillance and exploration operations to support UGV units have been developed. These surveillance and exploration algorithms can incorporate realistic constraints on platforms and sensors because of the operation of hardware-in-the-loop UGVs. To date, the research team has successfully demonstrated the algorithms on a test bed incorporating a high-fidelity simulation model of a UH-60A Black Hawk helicopter known as U.S. Army GENHEL and Segway robotic systems.

Further research on the developed test bed will provide significant scientific and technical advancement in the cooperative control of autonomous systems. The availability of autonomous UAV-UGV teams capable of complex cooperative behavior will enable military and civilian personnel to execute highly complex missions effectively and remotely.

This paper is organized in six sections including the present one. The second section presents the overall architecture of the test bed and mission objectives, and the unmanned rotorcraft simulation is delineated in section three. Section four introduces the robotic systems and section five provides the implementation aspects and preliminary results. The paper is summarized and concluded in section six.

II. Overall Architecture and Mission Objectives

The C^4ISR test bed aims to simulate real-time autonomous operation and coordination of air and ground units under certain mission objectives. The simulation test bed is built upon a high fidelity simulation model of a rotorcraft and hardware-in-the-loop emulation of robotic systems.

The architecture of the test bed is provided in Figure 1. The architecture is divided into three components based on the logical perception of the systems and considering the physical locations of the blocks. The Rotorcraft component is to simulate the RUAV operations of the test bed whereas the Robots component is used for UGV emulation. The Command and Control (C&C) component receives the abstracted information from other components and serves for decision making based on the overall mission objectives and coordination of RUAV and UGV units. The communications between the blocks and different components are through Ethernet and wireless networks depending on the data size and network traffic requirements.

The Rotorcraft component employs the nonlinear GENHEL simulation model of a UH-60 helicopter and accompanying continuously varying controllers together with a local discrete event supervisory (DES) controller (also denoted as “supervisor”). The details of the model and controllers will be described in later sections. The GENHEL model and its (continuous and DES) controllers are simulated in one computer. Although they could be implemented as a single program, the supervisor and the GENHEL model (with continuous controllers) are disjoined and talk to each other through socket communication. This method is preferred to separate the discrete event and continuous dynamics of the system in a modular structure. Similarly, the Robots component emulates the dynamics of UGVs using several networked Segway robots. When emulating many UGVs, it is also possible to use simulated Segway robots due to the limited number of actual robots, hardware limitations, and restricted availability of laboratory space. For this purpose, system identification is performed for the governing dynamics of the robotic system, and the identified dynamics are utilized as if the actual hardware is operating, which will be explained later. Moreover, a separate collision
avoidance algorithm continually supports the autonomous operation of the robotic systems, since utilization of actual hardware brings the danger of collision and necessitates an avoidance algorithm.

In the formulation of intelligent decision-making policies, the supervision of the entire fleet is managed by a chain of multi-tier discrete-event supervisors that are optimized at each level of operation, while the interactions between the supervisors in the hierarchy are obtained from the relationship between the discrete events and continuously-varying sensory information. The C&C component of the test bed obtains the information from the local supervisors of the RUAV and UGVs through a wireless network. The C&C computer receives constant updates on the current location of each vehicle, selects a desired path for each vehicle based on the current mission status, known enemy positions and expected behavioral patterns of enemy and friendly units, and issues corresponding waypoint commands. The C&C computer also determines what objects, such as enemy vehicles or buildings, each vehicle can see based on its position in the virtual (and/or actual) world via a line-of-sight algorithm.

In this study, the C4ISR system is used to manage a cooperative autonomous mission that requires intelligent control and decision-making. It is assumed that a highly maneuverable rotorcraft is used to search for enemy units and targets and to coordinate the operation of available ground units, while attempting to avoid detection and enemy fire. Judicious distribution of limited resources to accomplish the mission objectives has utmost importance to the mission. The proposed C4ISR system undertakes the task of mission management from the top level discrete-event supervision (DES) to the bottom level continuous-time regulation. The goal of the supervisory decision and control system is to autonomously achieve predefined mission objectives using a hierarchical structure of: (i) continuous-time control at the lower level for high precision maneuvering; and (ii) discrete-event supervisory control at the upper level for intelligent decision-making.

III. Unmanned Rotorcraft Simulation

A. Flight Dynamics Model

The RUAV component of the test bed is realized using a networked computer simulation with one computer running the flight dynamics model and another computer running a visualization of the rotorcraft flying over virtual terrain. The RUAV flight dynamics are simulated by a high-fidelity nonlinear model of a UH-60A Black Hawk helicopter known as U.S. Army GENHEL. The GENHEL simulation code is widely used by industry and the U.S. government and is accepted as a validated engineering model for handling qualities analysis and flight control design. The code models non-linear aerodynamic effects, and includes fuselage rigid body dynamics, rotor blade flapping and lagging dynamics, rotor inflow dynamics, engine/fuel control dynamics, actuators, and a model of the existing UH-60A automatic flight control system (AFCS). Modifications to the code allow for the replacement of the existing AFCS channels by the controllers presented in this paper, as well as networked communication (using UDP sockets) with the C&C and rotorcraft visualization computers.
B. Visualization

Visualization of the simulated RUAV is performed by an open source flight simulation program called Flight-Gear; although FlightGear is capable of simulating flight dynamics, its role in this paper is limited solely to the visualization of data provided by an external simulation. The features of FlightGear relevant to this paper include accurate terrain data for the entire Earth and the ability to place objects (e.g., buildings, bridges, and static vehicles that may be used as enemy targets) within the terrain. A multiplayer feature allows simultaneous operation of multiple vehicles that interact within the same virtual world.

The RUAV and the UGVs are linked in two ways: the C&C computer communicates with each vehicle to issue commands and receive position data and each vehicle has its own instance of FlightGear which communicates with every other instance of FlightGear so that all of the vehicles can be visualized in the virtual world. The network communication with the C&C computer and among the separate instances of FlightGear is performed using UDP sockets.

In addition to coordinating the network communication, each of the instances of FlightGear as well as the C&C computer must have the same data regarding terrain and placement of static objects (the positioning of dynamic objects is handled by the continuous network communication). After the location of each of the static objects is determined as described above, this information must be transmitted to each instance of FlightGear and the C&C computer to ensure that the virtual world is identical in each of its representations in the test bed.

1. Terrain Database Generation

Two important features of the C4ISR simulation test bed require the terrain profile of the location where the missions are performed. The line-of-sight algorithm, which will be described later, uses the terrain information to check whether an object in the sensor field is visible or masked. The altitudes between the object and rotorcraft are needed for this algorithm work. Moreover, the terrain database is needed to model the sensors required for a terrain following controller used on the RUAV.

A terrain database was extracted for the State College, Pennsylvania area, where the University Park campus of the Pennsylvania State University is located and the simulated mission will be executed. In order to produce a terrain database suitable for the purposes of this research, the elevations over an 80000 ft × 80000 ft area were extracted from FlightGear with a resolution of 20 ft and stored so that it can be easily accessed.

The total number of data points is \(4001 \times 4001 = 16008001\). The terrain profile of State College obtained from FlightGear and a topological map of the same area is given in Figure 2.

It is important to store this data in the most compact way to facilitate loading and processing the data. For this purpose, although the collected altitude data is of type \texttt{double}, which is used for floating numbers and binds 8 bytes, the data is stored in a binary file and as \texttt{unsigned short int} type, which takes 2 bytes (or 16 bits) of space on the hard disk. However, this means that the data that must be saved as integers in the range \(0 \rightarrow 2^{16} - 1 = 65535\). This difficulty can be circumvented by storing one decimal precision only (before saving: multiply the \texttt{double} data by 10 and convert to \texttt{unsigned short int}, before using in the simulation:...
convert unsigned short int to double and divide by 10), which is sufficient for both controller performance and line-of-sight algorithm. Therefore, the total data file size on the hard disk is 16008001 × 2 = 32016002 bytes ≈ 30.5 MB. The data can be loaded in a separate thread without interfering with the execution of the simulation or each time before the simulation starts. The average loading time is found to be less than 4 seconds for a computer with a 2.53 GHz Pentium 4 processor and 1 GB of RAM.

2. Object Placement

As described above, FlightGear allows for the placement of static objects such as airports, buildings, vehicles, etc. in the terrain. For this research, a number of vehicles are also added to the terrain to act as enemy ground targets. The objects that are always stationary (airports, buildings, etc.) are kept in the same location every time a mission is simulated. The enemy ground vehicles, which would normally move while requiring additional computers for their simulation that causes an increase in complexity, are placed in different locations for each simulation run. This is accomplished by another dummy program that randomly selects a latitude and longitude within the simulation area for each enemy ground vehicle. The program then uses the terrain database to find the elevation of the ground at each location, so that the enemy ground vehicles are positioned on the ground, not above or below. Finally, the program stores the positions of the enemy ground vehicles in two formats: one for use with FlightGear, so that the enemy ground vehicle is displayed in the visualization, and one for use by the C&C computer, which is used to determine when an enemy ground vehicle is detected by the RUAV or the UGVs.

3. Multiple Vehicle Visualization

In addition to displaying static objects, FlightGear is capable of networked multiplayer operation. This allows the visualization of the RUAV to include the moving UGVs. The structure of FlightGear requires that each moving vehicle run its own instance of FlightGear that communicates with every other instances; therefore, the RUAV and each UGV (whether it is simulated or corresponds to a physical robot) must have an instance of FlightGear that receives position data for its vehicle from either GENHEL, the Segway robot or the Segway robot simulation. Each of these instances then receives the positions of the other vehicles via network communication and displays them if they should be seen. The instances of FlightGear associated with the UGVs are also used to keep the visualizations of the UGVs on the ground in the virtual world by assigning the altitude of the UGV as ground level.

C. Simulated Sensor and Line-of-Sight

One of the most important elements in coordination of unmanned rotorcraft and ground vehicles is development of a simulated sensor for the RUAV. The simulated sensor replaces the actual sensor inputs in the simulation test bed, where actual data and/or visual detection is not available.

Despite the recent advancements towards the graphic visualization of enemy and friendly vehicles in the simulation environment, the extraction of information regarding the location of the static or moving targets from the computer screen is not usually available. The information flow in the simulation is oftentimes one way such that the objects are placed at given locations in the simulation, but the unknown locations of objects cannot be obtained from the visual data of the screen.

The usual sensor field is defined by an angle measured from the roll axis of the aircraft. There is a maximum range of the sensor readings\(^{10}\) which in many cases would exceed the maximum dimension of the simulated domain (80000√2 ft ≈ 21.5 miles). However, for this study it is assumed that the sensor has a 10000 ft maximum range of visibility for simulated missions. The location and orientation of the rotorcraft is given by the quadruple \((x_r, y_r, z_r, \psi)\) where the angle \(\psi\) is measured clockwise from +\(y\) to +\(x\), and the location of the target is given by the triple \((x_t, y_t, z_t)\), where \(z\) is the altitude above the sea level.

The simulated sensor is assumed to be located on the nose of the rotorcraft, aiming towards the ground with an angle \(\theta\) as measured from the vertical. The sensor visibility angle is denoted by \(\gamma\) as seen in Figure 3(a). The same figure also shows the distances that are necessary in the sensor field computation, such as the distance \(M\) between the rotorcraft and the point where the hypothetical sensor line intersects the ground, \((x_0, y_0)\). With the assumption that the terrain altitude does not change drastically inside the sensor field,
the following equations approximately hold.

\[ M = h \tan(\theta) \]  
\[ R_1 = h[\tan(\theta + \gamma/2) - \tan(\theta)] \]  
\[ R_2 = h[\tan(\theta) - \tan(\theta - \gamma/2)] \]  
\[ R_3 = h \tan(\gamma/2) \]

where \( h \) is the altitude of the rotorcraft above the ground, which is calculated from the absolute altitude \( z_r \) and altitude of ground level.

Figure 3(b) depicts a bird’s-eye view of the simulated sensor. The point \((x_0, y_0)\) and the relative position of the target to this point can be computed using the rotation formulas\(^1\) and the equations given below:

\[ x_0 = x_r + M \cos(\psi) \]  
\[ y_0 = y_r + M \sin(\psi) \]  
\[ x_t^0 = (x_t - x_0) \cos(\psi) + (y_t - y_0) \sin(\psi) + x_0 \]  
\[ y_t^0 = (x_t - x_0) \sin(\psi) - (y_t - y_0) \cos(\psi) + y_0 \]

The shaded region is the field of the simulated sensor in Figure 3(b) which can be approximated by two elliptical surfaces centered about the point \((x_0, y_0)\). The equations of these ellipses are given as:

\[ \text{Ellipse} 1 = \frac{(x_t^2 - x_0^2)}{R_1^2} + \frac{(y_t^2 - y_0^2)}{R_2^2} \]  
\[ \text{Ellipse} 2 = \frac{(x_t^2 - x_0^2)}{R_3^2} + \frac{(y_t^2 - y_0^2)}{R_2^2} \]

where the axes are given by the pairs \((R_1, R_2)\) and \((R_3, R_2)\) respectively. The simulated sensor confirms if the location of the target lies inside the shaded field by checking both ellipse equations.

The visibility of an object from the rotorcraft does not depend only on the field of the simulated sensor, i.e. being inside the field of the sensor is not enough to be seen, although this is a necessary condition. The other necessary condition is that the object in the field of the sensor must not be masked by the terrain. To ensure this, the line-of-sight algorithm checks the altitudes of points in between the rotorcraft and the target with a resolution of 20 ft \( \times \) 20 ft.

As seen in Figure 3 the line-of-sight algorithm checks the altitude of the points on the hypothetical line between the rotorcraft and the target. Note that this line lies in three dimensional space and is defined by two slopes. For the object to be seen, every point along that line should be clear of the ground. This test is performed recursively starting from the location of the target and moving towards the rotorcraft.

---

\( \gamma \)

\( \theta \)

\( x_r \)

\( y_r \)

\( z_r \)

\( R_1 \)

\( R_2 \)

\( R_3 \)

\( \psi \)

\( x_0 \)

\( y_0 \)

\( x_t^0 \)

\( y_t^0 \)

\( h \)

\( M \)

\( x_r \)

\( y_r \)

\( z_r \)

\( \theta \)

\( \gamma \)

\( R_1 \)

\( R_2 \)

\( R_3 \)

\( \psi \)

\( x_0 \)

\( y_0 \)

\( x_t^0 \)

\( y_t^0 \)

\( h \)

\( M \)

\( x_r \)

\( y_r \)

\( z_r \)

\( \theta \)

\( \gamma \)

\( R_1 \)

\( R_2 \)

\( R_3 \)

\( \psi \)

\( x_0 \)

\( y_0 \)

\( x_t^0 \)

\( y_t^0 \)

\( h \)

\( M \)
D. Controller Architecture

1. Low-Level - Longitudinal and Lateral-Directional Controllers

Both the longitudinal and lateral-directional controllers for the RUAV are based on previous research in control design. The longitudinal controller, originally designed to minimize transmission damage, uses an explicit model-following control scheme with a feedback portion designed using an LQR solution. The LQR gains are scheduled with airspeed to account for variations in the plant model. As illustrated in Figure 5(a), the controller is designed to regulate rotor speed, minimize transients in engine torque, and follow commands in vertical speed and pitch attitude. The lateral-directional controller, shown in Figure 5(b), also uses model-following with feedback designed using an LQR solution scheduled with airspeed. Inputs to the lateral-directional controller for high speed (above 30 knots) are commanded roll attitude and lateral acceleration for coordinated flight; for low speed, the commands are roll attitude and yaw rate. In this application, the yaw rate command is obtained by passing a heading command through a proportional-integral (PI) controller.

\[ \psi_{err} = \psi_{cmd} - \psi \]  
\[ r_{b,cmd} = K_P \psi_{err} + K_I \int \psi_{err} \, dt \]  

(a) Longitudinal controller  
(b) Lateral-directional controller

Figure 5. RUAV controllers

2. Mid-Level - Terrain Following, Airspeed Regulation and Waypoint Navigation

Terrain following is accomplished by converting AGL commands from the discrete-event supervisor of the RUAV into vertical speed commands used by the low-level longitudinal controller using a proportional-derivative (PD) filter. The vertical speed commands are limited to prevent the rotorcraft from entering an autorotative state (descent rate limit) and to prevent violating the continuous engine torque limit of the rotorcraft (climb rate limit); as with the low-level controllers, the vertical speed limits are scheduled with airspeed. Airspeed regulation converts the total airspeed command from the discrete-event supervisor of the RUAV into a pitch attitude command via a proportional-integral controller.

\[ h_{err} = h_{cmd} - h \]  
\[ V_{D,cmd} = K_P h_{err} + K_D \dot{h}_{err} \]  
\[ V_{err} = V_{cmd} - V \]  
\[ \theta_{cmd} = K_P V_{err} + K_I \int V_{err} \, dt \]  

Waypoint navigation is based on commands from the global discrete-event supervisor; specifically, the global DES provides a waypoint location (x-y position), a navigation mode command, and a heading command which is used only at low speed. The waypoint navigation system then directs the rotorcraft to the
given waypoint; the path the rotorcraft travels and what it does when it reaches the waypoint depends on the navigation mode. There are five navigation modes:

1. Fly in a direct path to the next waypoint and continue on to the following waypoint at the commanded airspeed

2. Fly along the path from the previous waypoint to the next waypoint and continue on to the following waypoint at the commanded airspeed

3. Fly directly toward the next waypoint at the commanded airspeed, then enter a figure-8 orbit around the waypoint at the minimum-power airspeed (shown in Figure 6(a))

4. Fly directly toward the next waypoint at the commanded airspeed, then enter a circular orbit around the waypoint at the minimum-power airspeed (shown in Figure 6(b))

5. Fly directly toward the next waypoint at the commanded airspeed, then hover at the waypoint with a given heading

For the first two modes, as well as the waypoint approach portion of the other modes, the waypoint navigation system produces a roll attitude command that is directly proportional to the error between the current flightpath angle of the RUAV and the next desired waypoint. This roll command has the effect of turning the RUAV toward the waypoint. The figure-8 and circular orbits are produced by introducing a number of intermediate waypoints to establish the shape of the orbit, which the RUAV navigates using roll angle commands as described above.

For hover mode, the RUAV switches from the model-following controllers described above to proportional-integral-derivative (PID) controllers with command filters, which are used to convert longitudinal position into a pitch angle command and lateral position into a roll angle command. The desired heading of the RUAV is converted into a yaw rate command through a PI controller.14

3. High-Level - Local Discrete Event Supervisor for Rotorcraft

Discrete Event Systems form the class of dynamical systems where the evolution of system dynamics in time depends on the complex interactions of various discrete events, such as initiation or completion of battlefield operations and success or failure of certain reconnaissance or surveillance missions. An FSA, or finite state automaton, is a model of behavior composed of states and transitions, and is used as one of several mathematical representations of discrete event systems. FSAs generate and recognize only the set of regular languages. Consequently, discrete event dynamical behavior of physical plants (e.g., rotorcraft and gas turbine engines) is often modeled as regular languages that can be realized by finite state automata.3
The local discrete event supervisor for rotorcraft maneuvers is used to decide on the set points for the continuous controllers depending on the operation and mission objectives. An FSA model of the rotorcraft is created considering the requirements to accomplish the mission and maneuvering capability of the rotorcraft. The local supervisor restricts the rotorcraft behavior while making the optimal trade-off such that the maneuver is completed with minimal effort and mission objectives are still satisfied. For example, the supervisor limits the rotorcraft in high speed and low altitude operation in the presence of enemy units by issuing the set points of the speed and altitude for the lower level controllers.

4. High-Level - Global Discrete Event Supervisor for C&C

The objective of the global discrete-event supervisor for command and control is to restrict the controlled plant behavior by disabling or enabling certain controllable events based on observed event strings and optimal control policy. The events are generated from the information of local supervisors, mission objectives and simulated sensor information. The supervisor selects a desired path for each vehicle based on the current mission status, known enemy positions and expected behavioral patterns of enemy and friendly units, and issues corresponding waypoint commands for the rotorcraft and the UGV units. The aim is judicious distribution of limited resources towards the goals of mission, which has utmost importance for mission management, therefore to increase the likelihood of accomplishing the mission objectives successfully.

The optimal control policy determines the expected long term behavior of the finite state model of the C&C and tries to maximize the performance of this behavior by selectively disabling or enabling certain controllable events. At each state of the operation, the supervisor selects the states that is not allowed to go rather then selecting a single state that will be forced to go. This approach is especially useful to increase the options that the C&C system may choose while the expected performance is still maximized. Moreover, there usually exits a cost associated with disabling an event (e.g. halting the mission during execution phase has a cost since the resources are already dispatched for the mission). Therefore, minimizing the restricted behavior (rather than forcing one path of operation) is also inherently aimed in the optimal control policy.

IV. Robotic Systems

A. Segway Robotic System

Four Segway robotic mobile platforms (RMPs) of the Networked Robotic and Sensor Intelligence Lab at Penn State have been used as a simulation for unmanned ground vehicles (UGVs). Each of the Segway RMPs is equipped with a SICK LMS200 laser range finder, six Devantech SRF05 sonars and six Sharp GP2D12 infrared sensors for obstacle avoidance. The laser range finder with a range of 80 m and resolution of the order of millimeters is used for both long range and short range obstacle avoidance. The sonar range finder with a range of 80 m and resolution of the order of millimeters is used for both long range and short range obstacle avoidance. Three sonars in the front and three in the back are used to provide short range obstacle avoidance information with a range of about 3 meters. The IRs are placed on the sides to enable the robot to see if it is safe to make turns. The sonars and IRs are controlled by a dedicated network of Brainstem General Purpose (GP) modules based on the PIC18C252 micro-controller. The brain of the entire system is a Dell Latitude D410 laptop which is connected to the Brainstem Network and Sick Laser and the RMP’s on board controller through USB. The robot itself is connected to the rest of the network through the use of IEEE 802.11b/g wireless network.

B. System Identification of Segway

This section describes the modeling of a Segway RMP. The model was obtained by using the system identification toolbox of MATLAB. The robot was excited using a pseudo random input in its typical work environment and the measured output of the onboard sensors was recorded. The input was constrained as follows

$$1 \text{ sec} \leq t_d \leq 6 \text{ sec} \quad (17)$$

$$-0.8 \text{ m/s} \leq v \leq 0.8 \text{ m/s} \quad (18)$$

$$-0.4 \text{ rad/s} \leq \omega \leq 0.4 \text{ rad/s} \quad (19)$$

$v$ and $\omega$ are the linear and angular velocities of the robot which are held constant for a period of $t_d$ seconds. Since the Segway has the dynamics of an inverted pendulum, an onboard balancing controller is always active.
which introduces high frequency noise to the response. The subspace method for system identification\textsuperscript{15} is performed to eliminate the effects of this noise. To choose the model order we follow standard techniques such as the Akaike’s Information Criterion (AIC).\textsuperscript{15} A fourth order model was found to be a good balance between prediction error and modeling complexity as shown in Figure [7]. The resulting state space model with the state vector $x = [v \quad \dot{v} \quad \omega \quad \dot{\omega}]^T$ is given by

$$x_{n+1} = Ax_n + Bu_n$$
$$y_n = Cx_n + Du_n$$

(20a) (20b)

where the matrices $A, B, C, D$ are as follows:

$$A = \begin{bmatrix} 0.96243 & -0.026344 & 0.11904 & 0.01982 \\ -0.029337 & 0.61297 & 0.024228 & 0.21747 \\ -0.12771 & -0.079849 & 0.76951 & -0.25586 \\ -0.0052022 & -0.47935 & -0.217 & -0.4911 \end{bmatrix}$$

$$B = \begin{bmatrix} -0.0081757 & -0.00027395 & 0.001303 & 0.085203 \\ 0.61297 & 0.024228 & 0.40611 & 0.080659 \\ -0.0052022 & -0.47935 & -0.217 & -0.4911 \end{bmatrix}$$

$$C = \begin{bmatrix} 5.3435 & -0.39613 & -0.14567 & 0.092586 \\ 0.12038 & 1.4111 & 0.0012965 & -0.2697 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 & 0 \end{bmatrix}$$

Figure [7] shows the input, measured response and the response of the 4\textsuperscript{th} order model to this input.
C. Robot Simulation

The Stage simulator of the Player/Stage project has been used to simulate the Segway robots. It is a 2D simulator capable of simulating sensors like laser, sonar, camera, pan-tilt-zoom (ptz), and has simple friction models for simulating built-in objects. Stage is capable of simulating a large population of robots with low-fidelity models. To more realistically simulate the dynamics, the model of the Segway obtained in section has been used in conjunction with Stage as shown in Figure 9. Each robot in the Stage simulator runs an instance of Player robot server to provide access to sensors and actuators to the clients.

The Player is the lowest level functional block in the hierarchy which accesses to the robots’ physical hardware such as laser, sonar and the actuators, and is usually run on the onboard computer. The Action Generator is the block that consists a set of fundamental behaviors that the robot can perform. More sophisticated behaviors are built upon the fundamental ones which are Search, Approach, Idle, Ignore, GoTo and Planner. In the context of this paper, Idle (robot does nothing), GoTo (robot goes to a waypoint) and Planner (robot plans the path from current position to any waypoint and uses Floyd-Warshall algorithm for shortest path) actions are employed. The Event Generator is an observer that implements continuous to discrete-event conversion.

V. Preliminary Results

The results presented herein demonstrate the seamless integration of rotorcraft simulation and robot simulation. The goal is to verify the benefits of cooperative control of heterogeneous agents such as RUAV and UGV in enemy environment. The experiment is defined as a supply mission of the UGV. The UGV starts from the base and aims to reach to the target point which lays in enemy territory. The shortest path seems to be the best strategy of the UGV if there is no enemy unit between base and target and the UGV follows its best strategy under limited information. However, the enemy unit can destroy the UGV before its presence can be detected by UGV sensors. The RUAV has a better sensing capability than UGV and can search possible enemy locations before the UGV approaches those points. If the enemy is detected by the RUAV, the local strategy of the UGV is overwritten by higher-level C&C controller and UGV follows a longer but safer route to the target.

Figure 10. Spatial navigation of RUAV and UGV in the presence and absence of enemy threat
Figure 11. Temporal navigation of RUAV and UGV in the presence and absence of enemy threat

Figure 11 shows the mission map for the operations with and without an enemy threat. The solid line is the path of the RUAV while searching the possible enemy locations in the map. The dashed line is the path of the UGV when rotorcraft does not detect enemy presence and the dotted line is the path when rotorcraft detects the enemy by its sensor. The shaded region is the fire range on the enemy that UGV has to avoid. Figure 11 exhibits the behavior of RUAV and UGV for same mission executions in time domain.

The robot and rotorcraft begin the mission at the same time. When simulation starts, the UGV initiates its GoTo behavior while the RUAV searches for possible enemy locations as seen in Figure 10. From the simulated sensor data of the RUAV, either “enemy sighted” or “path cleared” event is generated. Based on the generated event, the UGV may proceed with its original path or deviate for a longer but safer path. Both RUAV and UGV rendezvous at the target location at the end of the simulation. It is seen in Figure 11 that behaviors of UGV with and without enemy presence differs at 300 seconds, when the enemy is detected by the RUAV, although RUAV passes over the actual location of the enemy 25 seconds later.

VI. Summary and Future Work

A comprehensive simulation testbed has been developed for validation of various control strategies on coordinated and cooperative control of RUAV and UGV operations. The testbed aims to incorporate high-fidelity simulation model of a rotorcraft and hardware of robotic mobile platforms to establish a framework for C³ISR systems. The seamless integration of the RUAV component and simulation of a single robot has been achieved to successfully demonstrate the ability of continuous-domain lower level and discrete-event-domain higher level controllers. The future work includes:

- Integration of many UAVs and UGVs for decision making and intelligent control of C³ISR operations;
- Implementation of the actual hardware of robotic systems for mission execution;
- Intelligent decision and control of distributed autonomous systems, where each component has its own local control;
- Decision making and mission planning modifications through a high-level C&C coordinator;
- Incorporation of optimal DES control laws for enhanced mission management;
- Development of a flexible software architecture to employ arbitrary models and controller designs.
References