A Hardware Testbed for Multi-UAV Collaborative Ground Convoy Protection in Dynamic Environments

Philip Twu, Rahul Chipalkatty, Jean-Pierre de la Croix, Thiagarajan Ramachandran, Jeremy Shively, Magnus Egerstedt

Georgia Institute of Technology, Atlanta, GA 30332, USA

Amir Rahmani
University of Miami, Coral Gables, FL 33146, USA

Ryan Young
Rockwell Collins Inc., Cedar Rapids, IA 52498, USA

In this paper, we will present a hardware testbed for multi-UAV systems that bridges the gap between algorithm design and field deployment. The testbed allows for UAV coordination algorithms, that have been shown to work in simulation, to be further tested in an environment where limited on-board computational resources, wireless communication constraints, environmental noise, and differences in the UAVs modeled versus actual dynamics come into effect. In particular, we will introduce an efficient assignment algorithm. This algorithm is used in a multi-UAV ground convoy protection scenario, where UAVs escort the ground convoy and are deployed to check potential threats along the way.

I. Introduction

In the design of future unmanned vehicle systems, coordination of heterogeneous teams is one of the key issues that must be resolved. Importantly, unmanned ground vehicles (UGVs) offer utility when it comes to local surveillance, logistics, and transportation tasks. Unmanned air vehicles (UAVs) offer utility in large-scale surveillance, scouting, and protection tasks. However, the differences between the UAVs’ and UGVs’ sensing capabilities and dynamic characteristics pose significant challenges when designing, tasking, and piloting these autonomous systems.

The overarching problem under consideration here is the issue of how the UAVs should be assigned to solve dynamic tasks while ensuring that certain coordinated ground-air specifications are met. In particular, we focus on the convoy protection problem as an instantiation of this overarching theme. What this entails is that a collection of UAVs must provide protection to a convoy of UGVs in the sense that (1) at least one UAV should always be placed above the convoy so as to provide protection from potential immediate threats, and (2) pop-up threats along the path traversed by the convoy must be investigated and cleared before the convoy arrives, which implies that UAVs must be dispatched away from the convoy to handle these threats.

Emails:
Email: {ptwu, chipalkatty, jdelacroix, tramachandran3, jshively3}@gatech.edu, magnus@ece.gatech.edu
Email: amir@teamrahmani.com
Email: syoung@rockwellcollins.com
One main contribution of this paper is an algorithm that assigns and dispatches UAVs to threats while ensuring that convoy protection is maintained for all times. This assignment must moreover be done in such a way that the fuel consumption is balanced between the UAVs so that the time in-between having to refuel is maximized. As an additional requirement, we also must be able to run the algorithms on a platform with limited computational power over communication channels with limited capacities.

A hardware platform consisting of mobile robots as UGVs, and quadrotor craft as UAVs is developed to highlight robustness properties as well as performance of the aforementioned assignment algorithm in resource constrained environments. In the following sections, we will define the algorithms and hardware interfaces to be deployed.

II. The Assignment Problem

A. Complexity Considerations

Given a single UAV and $N$ potential threats that must be cleared, together with a cost associated with traversing between the threats, the problem of selecting the order in which the threats should be visited (and cleared) is a variant of the well-studied traveling sales person (TSP) problem. Unfortunately, TSP is a NP-hard problem, i.e., there are no polynomial (in the number of threats) time algorithms that can solve the problem. As a consequence, any attempt at tackling the full-fledged TSP using limited computational resources is doomed to failure. The reason for this computational intractability is that, in order to solve the TSP problem, one has to establish the order in which the threats are visited. If, on the other hand, this order is a priori given, the problem would already be solved.

If there are multiple UAVs (as is the case in the convoy protection scenario under consideration here), the complexity of the TSP problem is reduced if the number of UAVs is greater than or equal to the number of threats. The reason for this is that we end up with a so-called matching problem, where we simply have to decide which threat should be visited by which UAV. And, since there are no threats left unvisited after all UAVs have visited one threat, there is again no ordering of the threats that must be decided upon. The matching problem can be solved in cubic time (in the number of threats) using the Hungarian algorithm, as is done, for example, in works of Ji et al. and Moore and Passino. If there are fewer UAVs than threats then the NP-hardness result still applies, and this is the situation (fewer UAVs than threats) that we can expect.

Numerous algorithms for approximating and addressing the TSP problem in a computationally tractable manner have been proposed. These algorithms typically fall under two categories, namely (1) greedy algorithms, where each agent is trying to locally maximize and minimize a cost based solely on the instantaneous cost rather than over the whole time horizon, and (2) auction-based and game theoretic algorithms, where the agents get to “bid” on which tasks they want to be assigned to. In the greedy case, the solution is clearly suboptimal, even though it was shown in that asymptotically (as the number of threats approach infinity), the solution gets to within a constant factor of the optimal solution. Auction-based solutions, on the other hand, typically require a significant amount of information to be shared by the agents and, despite this increase in shared information, the inherent complexity of the problem remains the same. Solutions that fall in-between these two approaches were presented in, where decentralized suboptimal solutions where obtained by focusing on the feasibility of the solution over limited time horizons rather than the minimization of any particular cost function, and in, where the assignment and allocation problems were addressed for more specialized surveillance missions. This last approach (as exemplified by) is one that we will follow in this work in that the special structure of the convoy protection problem will be leveraged to obtain good
suboptimal solutions to an intractable problem.

The convoy protection problem is not a standard TSP (single or multi-UAV) problem even though it shares some of the same features. The two main differences are:

- **Threat dynamics:** The threats are not given all at once. Instead they show up throughout the mission, and as a consequence, the problem must be resolved whenever a new threat appears.

- **Overall cost:** The cost to be minimized is not the total distance traveled or the total fuel consumed by the team of UAVs. Instead it needs to reflect the fact that the fuel consumption should be balanced among the different UAVs. As such, it is a minmax cost in that what must be minimized is the fuel consumption of the UAV who has to expend the most fuel.

As a consequence of these considerations, the convoy protection problem is still NP-hard and in order to be able to still address it in a resource constrained environment, simplifying assumptions must be made. However, as opposed to the discussed related work, we will take the explicit structure of the convoy protection problem into account when making these assumptions. The solution will be suboptimal, by necessity, and the quality of the solution will depend on how reasonable these assumptions are. In the next section, we will discuss the main assumptions needed to develop our convoy protection algorithms.

B. **Simplifying Assumptions**

We here outline the assumptions we make to take an inherently intractable problem and make it computationally tractable. These assumptions must be reasonable in the sense that they both preserve the problem in that it should remain as close to the original convoy protection problem as possible, as well as ensure that the resulting algorithms are applicable to a large class of different types of unmanned vehicles.

The simplifying assumptions are:

**A1 Bounded number of threats:** At any given time, only a fixed number of threats will be considered. Even though we place no a priori upper limit on the number of threats that may appear, we will restrict our algorithm to only consider the closest $M$ threats, where $M$ is a fixed number. Without further assumptions, this will still make the complexity factorial in $M$, which may be a large number (depending on what $M$ is), but at least it is bounded.

**A2 One-dimensional threat distribution:** As the convoy is moving along an established route, e.g., a road, only threats on (or close to) the road will be considered. As such, the threats will be placed along a one-dimensional path and they will be encountered sequentially. The result is that the order in which the threats are cleared is already given in that the closest threat is cleared first.

**A3 Not all threats must be cleared:** The objective of the convoy protection algorithm is to ensure that the convoy is not encountering any threats. There are two ways in which this can be achieved. The first is by clearing threats, as previously discussed. However, we can also replan the path the convoy takes in order to produce a detour around a threat. This will be done if the threat cannot be cleared in time.

These three assumptions will allow us to produce a polynomial-time algorithm for solving the convoy protection problem. However, in order to do so, we also need to make two additional assumptions. These do not pertain to the
abstract problem formulation, per se, but rather to the underlying cost models used to evaluate the performance of the UAVs. The cost assumptions

**A4 Known costs:** Although fuel consumption is a function of a number of factors, including altitude and speed of the UAVs, we must be able to predict what it will cost to clear a threat. As such, we will, solely for the purpose of making these prediction, assume that the UAVs will move with a constant velocity and altitude when clearing the threats. This will not be the case during the actual clearing of the threats but we will assume that it suffices as a model for making predictions.

**A5 Ordered costs:** Underlying our algorithm is the key assumption that it will require less fuel to stay with the convoy than to be dispatched to clear a threat. As such, we will make this assumption in order to establish an order over the costs which will moreover enable us to define an effective convoy protection algorithm.

### III. The Convoy Protection Algorithm

This section discusses the technical details behind the proposed convoy protection algorithm. It is divided into three parts. The first, dealing with the mathematical formulation of the general problem, is where the goal is to choose the optimal UAV to dispatch in order to survey and clear upcoming threats so as to balance fuel consumption. The second part gives details on a realistic fuel model which will be used to describe the UAV fuel consumption, namely the Sikorsky UH-60 Black Hawk model. This model should be thought of as pertaining to one particular instantiation of aerial vehicles, but the assignment algorithm does not depend on this particular choice of fuel model for its operation. The third part details the actual convoy protection algorithm which will be used to solve the UAV assignment problem efficiently for the realistic fuel model.

### A. Problem Formulation

The assignment problem is one of the fundamental problems in combinatorial optimization. In the setting of a UAV team providing ground convoy protection to a number of UGVs, it is important to make quick decisions in real time so that the task is allocated to the right UAV and a certain cost objective is optimized. For the convoy protection mission scenario, the UGVs are traveling along a nominal path. The task of the UAVs is to provide protection to the ground vehicles, while clearing threats as they emerge in front of the path of the UGVs.

A natural choice for the cost objective is to balance the fuel consumptions of the UAVs. In a realistic convoy protection scenario, the fuel carried by each UAV is limited. By balancing the fuel consumption amongst the UAVs, the duration in which all UAVs remain operational is being maximized. Therefore, in this paper, we will use the maximum fuel consumption amongst the UAVs as the cost function in designing the scheduling and task allocation algorithm for the UAVs.

To formulate the problem in a mathematical setting, we need to first provide a model for the overall mission of the UGVs and the possible threats. Furthermore, since the convoy protection problem is highly related to the TSP problem, we, as already discussed, make a number of simplifying assumptions to lower the complexity of the problem. This is of paramount importance as the algorithm must run repeatedly on a computationally limited resource in order to respond to dynamic threats.

Assumption **A2**, simply says that the UGVs follow a set path. This is a realistic assumption as the UGV is expected
to follow roads or other easily traversable structures in the environment. Along the nominal convoy path, there are a number of threats that the UAVs must survey and possibly clear. We classify the threats into two categories. One category of threats are persistent. These are threats that the UGVs must avoid since they cannot be cleared by the UAVs. And, as per Assumption A3, we also want to allow for the possibility of threats not being cleared if they cannot be surveyed by a UAV in time. In fact, it is assumed that the UAV-UGV team can only detect the threats up to a distance $D$ away from the location of the UGVs and, as a consequence of Assumption A1, the threat density is such that no more than a given number $M$ of threats can appear within this distance at any given time. But, if a threat pops up in front of the UGVs within distance $D$ and if all available UAVs are dispatched beyond the threat, clearing of the threat may not be possible. If this is the case, they will be treated as persistent threats. In both of these cases (inspected persistent threats and threats that can not be inspected in time) the path of the UGVs will be re-planned to avoid the threats.

The other category of threats are non-persistent, in which case the threats are removed after being checked by the UAVs. Regardless of the type, all threats must, if possible, be checked and surveyed by the UAVs and the threat category cannot be established without an UAV inspecting the location of the threat. If a UAV does not get to a threat in time, it will be treated as a persistent threat and, as such, force a replanning of the UGV convoy.

Figure 1. The assignment problem model
In order to formulate the assignment problem, we model the UGVs as a single point mass moving along a one-dimensional (not necessarily straight) path as shown in Figure 1. The UAVs are either flying alongside the UGVs (meaning they are at the same position along the path), or they can fly ahead of the UGVs. As they fly ahead of the UGVs, they do not have to follow the path. Instead, we will assume that they follow the Euclidean shortest distance to the target location. Hence, for this model, the motion of the UGVs is viewed as moving along a one-dimensional corridor which, as already discussed, allows us to establish an order among the threats which significantly cuts down on the complexity of the problem. At the same time, this is a realistic assumption that does not significantly limit the applicability of the proposed method.

With this model, we do not consider the kinematic constraints of the UAVs and UGVs, and the UAVs are assumed to be able to change direction and turn around while checking threats. At time \( t \), the position of the UGV team is denoted to be \( x_g(t) \). The positions of the individual UAVs are denoted as \( x_i(t) \), where \( i = \{1, 2, 3\} \) labels the UAVs.

We assume that the team can detect threats for a distance of \( D \) in front of the UGVs. Hence only threats contained in this window are known to the UGV-UAV team. At time \( t \), we assume that there are \( N(t) \) threats in this range. The location of threats are assumed to be fixed, and they are denoted as \( \tau_j, j = 1, 2, ..., N(t) \). The sequence of the known threats at time \( t \) are denoted as \( \tilde{\tau}(t) = [\tau_1, \tau_2, ..., \tau_{N(t)}] \). This known threat sequence has time-varying length \( N(t) \). \( N(t) \) increases when a pop-up threat is detected, and decreases when a threat is cleared. Note that all threats (pop-up or not) are in front of the UGVs on the intended path.

Furthermore, we denote the total number of threats as \( N \) (where we have, for notational convenience, suppressed the explicit dependence on \( t \), as will be done throughout), and the sequence of all threats is denoted as \( \tilde{\tau} = [\tau_1, \tau_2, ..., \tau_N] \). The overall mission of the UAVs is to clear all the threats (assuming none of them are hostile). If any of the threats are hostile, then the path taken by the UGVs is re-planned, the problem is re-initialized, and the assignment algorithm is restarted.

Naturally, when the UAV is checking a threat, it should fly with a higher speed than when it is cruising along with the UGVs. The velocity of the UGV team is assumed to be \( V_g \). We assume that when the UAVs are flying alongside the UGVs, they fly with the same speed \( V_g \). However, when they are assigned to clear a threat or coming back after clearing a threat, they fly with the speed \( V_a \). Throughout of this paper, we assume \( V_a > V_g \).

We denote the starting time of the overall mission as \( t_0 \), and the time when all \( N \) threats are cleared as \( T \). It should be noted that since we aim at developing a real-time algorithm to select and assign the UAVs to clear the threats, the only information available at time \( t \) is the \( N(t) \) threats within range \( D \), and our assignment algorithm only uses this information to make decisions.

The problem considered in this paper is to devise algorithms to dispatch the UAVs to clear all upcoming threats while balancing the fuel consumptions of the UAVs. The fuel consumption of \( i \)-th UAV at time \( t \) is denoted as \( f_i(t) \). \( f_i(t) \) is considered to be the state of the UAVs, and they are known. It should be noted that at the beginning of the mission, the fuel consumption of all the UAVs are 0, hence \( f_i(t_0) = 0, \forall i \). The details of how fuel consumption is calculated for the UAVs is given in the next section. With this information, we are ready to formulate the UAV assignment problem.

**Problem 1** Design an algorithm to assign UAVs such that the maximum fuel consumption of amongst the UAVs is minimized when all threats \( \tilde{\tau} \) are cleared. Hence, we aim to solve the following optimization problem:

\[
\min_{\Pi} \max_{j \in \{1, 2, 3\}} \{f_j(T)\},
\]

where \( \Pi \) is a assignment algorithm that maps from time to \( \{1, 2, 3\} \)
Remark 1 Even though we only consider three UAVs in the problem formulation, the algorithms and results presented in this paper can be extended to any number of UAVs.

B. Realistic UAV Fuel Model

The fuel consumption model for the UAVs will be based on fuel consumption data for the Sikorsky UH-60 Black Hawk, as shown in B. Although the algorithm is quite general and does not rely on any particular choice of fuel model, we do chose this particular model to highlight the realism associated with the assumptions made. As a consequence, the fuel consumption rate at any given moment will be a nonlinear function of the UAV’s speed, weight, and altitude. Therefore, the fuel consumption rate of the $i$th UAV at time $t$ is given by

$$\frac{df_i(t)}{dt} = Q(v(t), w(t), h(t)),$$

where $v(t), w(t), h(t)$ are the UAVs speed, payload weight, and altitude at time $t$, respectively.

![Figure 2. Fuel consumption model for UAVs based on data for the Sikorsky UH-60 Black Hawk (shown at varying altitudes and GTOW).](image)

As noted earlier, in our convoy protection scenario, the UAVs will, for the purpose of the convoy protection algorithm, be assumed to only fly at two speeds: $V_g$ when following the UGV convoy, and $V_a$ when surveying and clearing a threat. It is clear that this will not be the actual speeds used by the actual vehicles but, as the algorithm is constantly re-evaluated, the errors associated with the constant speed assumption will be compensated for. What is paramount though, is that the algorithm is able to predict reasonably well what the cost is, associated with different assignments, as per Assumption A4.

Similarly, for the purpose of evaluating the algorithm, the payload weight and altitude of the UAVs are assumed to remain constant $w(t) = w^*$ and $h(t) = h^*$ throughout the individual maneuvers needed to inspect threats. Note that this assumption is the same as Assumption A4 and it is only there in order to make the algorithm tractable. However, as the assignments are re-evaluated throughout the mission, the actual altitudes may very well change. Therefore, for the sake of computation, the fuel consumption rate of the $i$th UAV at time $t$ can be simplified to

$$\frac{df_i(t)}{dt} = \begin{cases} 
Q(V_g, w^*, h^*) & \text{if } v(t) = V_g \text{ (UAV } i \text{ is following convoy)}, \\
Q(V_a, w^*, h^*) & \text{if } v(t) = V_a \text{ (UAV } i \text{ is surveying/clearing threat)}. 
\end{cases}$$

(3)
A scaled version of this fuel model which will be used to describe fuel consumption amongst the UAVs. This scaling scheme simply translates and contracts the fuel models in Equation 3 to make them applicable to the physical parameters relevant to the quadrotor UAVs.

C. Selection policy and task allocation algorithm

In this section, we introduce our algorithm to assign the UAVs based on their current level of fuel consumption. In this algorithm, we use an optimization-based selection policy that selects (and assigns) one threat to one of the UAVs when an event triggers. Events that trigger the re-evaluation of the selection policy can be one of the following:

1. A threat is cleared by any of the assigned UAV.
2. A pop-up threat is detected.

Hence, the selection policy is only re-evaluated when the known threat sequence $\bar{\tau}(t)$ is modified. In order to allow real time execution of the algorithm, the optimization problem needed for the selection policy is chosen so that it can be solved quickly on-line. In order to reduce the problem from NP hard to polynomial complexity, we only consider the closest threat when an event triggers. This does not cause inefficiency in clearing multiple threats since each time a threat is cleared, the selection policy is re-evaluated. We denote the position of the closest threat to be $\tau \in \mathbb{R}^2$, hence

$$\tau = \min_{j \in \{1,2,3\}} \tau_j. \quad (4)$$

1. Selection Policy

In our framework, the selection policy is a function that maps time $t$ to the set of the UAVs $\{\{1,2,3\}\}$ in our case, and it is the solution of an optimization problem. This optimization policy is based on computing the fuel consumption associated with letting a UAV clear the target threat ($\tau$) and returning to the convoy. Even though this may actually not be what is done (as new threats are constantly reconsidered), the resulting algorithm minimizes the accumulated fuel consumption over all other assignments provided that the assigned UAV will return to the convoy after the threat is inspected. We take into account that the UGVs are moving constantly along the path with speed $V_g$, and that the closest threat $\tau$ may be a pop-up threat.

When an event triggers at time $t$, the following optimization problem is solved:

$$P_1(t) = \arg \min_{i \in \{1,2,3\}} \left\{ \max_{j \in \{1,2,3\}} \left\{ f_j(t) + \begin{cases} Q(V_a, w^*, h^*) T_i(t), & \text{if } j = i \\ Q(V_a, w^*, h^*) S_j(t) + Q(V_g, w^*, h^*) (T_i(t) - S_j(t)), & \text{if } j \neq i \end{cases} \right\} \right\}. \quad (5)$$

The optimization problem is a min-max problem. $T_i(t)$ represents the time for $i$-th UAV to clear the threat and return to the UGVs (with speed $V_a$). If the $j$-th UAV is not assigned, $S_j(t)$ represents the time for it takes to return to the UGVs (with speed $V_a$). $T_i(t) - S_j(t)$ therefore represents the time for the $j$-th UAV to fly alongside the UGVs with speed $V_g$ until the assigned UAV flies back. If the UAV $j$ is flying alongside the UGV at time $t$, then $x_j(t) = x_g(t)$ and $S_j(t) = 0$. Hence, the quantity

$$f_j(t) + \begin{cases} Q(V_a, w^*, h^*) T_i(t), & \text{if } j = i \\ Q(V_a, w^*, h^*) S_j(t) + Q(V_g, w^*, h^*) (T_i(t) - S_j(t)), & \text{if } j \neq i \end{cases} \quad (6)$$

represents the accumulated fuel consumption of the $j$-th UAV, assuming UAV $i$ is assigned.
It is important to note here that the convoy protection problem formulation, along with the corresponding task assignment algorithm presented in this section, is independent of how the quantities \( T_i(t) \) and \( S_j(t) \) are computed for estimating the flight times of the UAVs. Section 3 shows how to compute the two quantities when the path taken by the UGVs can be approximated as piecewise-linear and is defined by a sequence of waypoints.

As shown in Figure 3, \( P_1(t) \) assigns the UAV so that the UAV with most accumulated fuel consumed after the clearance of the threat is minimized.

Let \( M \) denotes the total number of UAVs (in this case \( M = 3 \)). Min-max problems are generally hard to solve, and the complexity of computing \( P_1(t) \) is \( O(M^2) \). However, under certain design choices of UAV speeds \( V_a \) and \( V_g \), weight \( w^* \), and altitude \( h^* \), we can reduce the optimization problem to one that has \( O(M) \) (linear) complexity. The assumption needed for this is that it is more costly in terms of fuel consumption to be inspecting a threat than to remain with the convoy, as per Assumption \( A5 \).

**Theorem 1** If UAV speeds \( V_a \) and \( V_g \), payload weight \( w^* \), and altitude \( h^* \) are chosen such that

\[
Q(V_a, w^*, h^*) > Q(V_g, w^*, h^*),
\]

then the optimization problem \( P_1(t) \) can be solved by solving another optimization problem:

\[
P(t) = \arg \min_{i \in \{1, 2, 3\}} \left[ f_i(t) + Q(V_a, w^*, h^*) T_i(t) \right].
\]

Hence, \( P(t) = P_1(t) \).

**Proof 1** For the inner, maximum part of the \( P_1(t) \), denote its solution by \( L(i, t) \) for a fixed \( i \).

\[
L(i, t) = \arg \max_{j \in \{1, 2, 3\}} \left\{ f_j(t) + \begin{cases} Q(V_a, w^*, h^*) T_i(t), & \text{if } j = i \\ Q(V_a, w^*, h^*) S_j(t) + Q(V_g, w^*, h^*) (T_i(t) - S_j(t)), & \text{if } j \neq i \end{cases} \right\}
\]

Thus

\[
P_1(t) = \arg \min_{i \in \{1, 2, 3\}} \{ L(i, t) \}.
\]

If \( L(i, t) = i \), \( \forall i \), then:

\[
P_1(t) = \arg \min_{i \in \{1, 2, 3\}} \left[ f_i(t) + Q(V_a, w^*, h^*) T_i(t) \right] = P(t).
\]
Note that since $Q(V_g, w^*, h^*) < Q(V_a, w^*, h^*)$,

$$Q(V_a, w^*, h^*)S_j(t) + Q(V_g, w^*, h^*)(T_j(t) - S_j(t)) < Q(V_a, w^*, h^*)T_j(t). \quad (12)$$

If $\exists i^*$ such that $L(i^*, t) \neq i^*$, this means that there is one UAV ($i^*$) such that, if it is assigned, the total fuel consumed by $i^*$ is less than all other $j \neq i^*$ even though it consumed more fuel than all others during this time ($T_j(t)$) because $Q(V_a, w^*, h^*)S_j(t) + Q(V_g, w^*, h^*)(T_j(t) - S_j(t)) < Q(V_a, w^*, h^*)T_j(t)$, $\forall i, j$. Hence in this case

$$i^* = \arg \min_{i \in \{1, 2, 3\}} \{f_i(t) + Q(V_a, w^*, h^*)T_i(t)\} = \arg \min_{i \in \{1, 2, 3\}} \{L(i, t)\}, \quad (13)$$

and therefore:

$$P_1(t) = P(t).$$

To describe in words, $P(t)$ assigns the UAV so that, after the threat is cleared and returned to the UGVs, the total fuel consumed for the assigned UAV is minimum over all the UAVs. The projected time for the clearance of the threat is $T_i(t)$, where $i$ denotes the assigned UAV. It should be noted that during this time, the UGVs has moved forward by $V_g T_i(t)$ distance.

2. Algorithm

Now we will present our task allocation algorithm which uses the above selection policy.

**Algorithm: Task Allocation Algorithm**

**Initialize:** Set $t_0$.

**Iterate:** until all threats are cleared

**If** an event is triggered:

1. update $\bar{\tau}(t) = [\tau_1, \tau_2, ..., \tau_{N(t)}]$.
2. find $\tau = \min_{j \in \{1, 2, ..., N(t)\}} T_j$.
3. Solve $i^* = P(t)$ and use the solution $i^*$ as the assignment to clear $\tau$.
4. wait until another event is triggered.

**Result:** a task allocation algorithm that clears all threats (if none are hostile) and solves problem 1.

3. **Computing UAV Flight Times**

In order to estimate the fuel consumed by each UAV for the task allocation algorithm, it is necessary to be able to predict how long it will take a UAV to visit a threat and to rendezvous with the UGV convoy. Here, we show how to compute the quantities $T_i(t)$ and $S_j(t)$ as seen in (5) and (8) for the special case when the UGV convoy’s path can be approximated as a 2-D piecewise-linear path as defined by a sequence of waypoints.
To compute the UAV flight times corresponding to when the UGV convoy takes a 2D piecewise-linear path, it is first necessary to compute the time it takes for the UAV to rendezvous with a convoy traveling in a straight line. Such a scenario is illustrated in Figure 4 and the time it takes for a UAV to complete such a task is summarized in Lemma 1.

**Lemma 1** Suppose at time $t_0$, a UAV is located at $y_a(t_0)$, while the UGV convoy is located at $y_g(t_0)$. Let the UGV convoy travel at a constant speed $v_g$ at an angle $\theta$ relative to the UAV’s initial position, while the UAV travels in a straight line at constant ground track speed $V_a$ to rendezvous with the convoy as shown in Figure 4. If the UAV is given a head start time of $\Delta t$ before the convoy, then $R(y_a(t_0), y_g(t_0), \theta, \Delta t)$, the time at which the rendezvous occurs after the convoy starts moving, is given by

$$R(y_a(t_0), y_g(t_0), \theta, \Delta t) = \begin{cases} \frac{-\beta + \sqrt{\beta^2 - 4\alpha \gamma}}{2\alpha}, & \text{if } V_a \Delta t \leq D \\ \text{Does Not Exist}, & \text{otherwise} \end{cases}$$

(14)

where

$$\alpha = V_a^2 - V_g^2$$

(15)

$$\beta = 2V_a^2 \Delta t + 2V_a D \cos(\theta)$$

(16)

$$\gamma = V_a^2 \Delta t^2 - D^2.$$  

(17)

**Proof 2** The time it takes to rendezvous is based only on the relative position and heading of the UAV with the convoy. Therefore, without loss of generality, assume that at time $t_0$, the initial position of the UGV convoy is given by $y_g(t_0) = (0, 0)$ and that it is traveling along the positive x-axis. Looking at Figure 4, we see that the corresponding initial position of the UAV is $y_a(t_0) = (D \cos(\theta), D \sin(\theta))$. If the UAV travels at an angle $\phi$ relative to the convoy’s path, then the positions as a function of time are given by

$$y_a(t_0 + t) = (D \cos(\theta) + V_a \cos(\phi)(t + \Delta t), D \sin(\theta) - V_a \sin(\phi)(t + \Delta t))$$

$$y_g(t_0 + t) = (V_g t, 0).$$

To have the UAV rendezvous with the convoy, we must then solve for the $t^*$ at which $y_a(t_0 + t^*) = y_g(t_0 + t^*)$. Therefore, equating the x and y coordinates of the two at time $t^*$ gives

$$\begin{align*}
V_a \cos(\phi)(t^* + \Delta t) &= V_g t^* - D \cos(\theta) \\
V_a \sin(\phi)(t^* + \Delta t) &= D \sin(\theta).
\end{align*}$$

Figure 4. Diagram used in Lemma 1 for computing the time it takes for a UAV to rendezvous with the UGV convoy traveling in a straight line.
After eliminating \( \phi \) by using the property \( \cos(\phi)^2 + \sin(\phi)^2 = 1 \), we arrive at a quadratic equation for \( t^* \) of the form

\[
\alpha(t^*)^2 + \beta t^* + \gamma = 0,
\]

where \( \alpha, \beta, \gamma \) are as given in (15). Notice that rendezvous is possible in non-negative time only when \( V_d \delta t \leq D \), so gamma \( \leq 0 \). Moreover, the initial assumption that \( V_a > V_g \) means that \( \alpha > 0 \). Therefore, \( \sqrt{\beta^2 - 4\alpha \gamma} \geq |\beta| \) and so the only non-negative solution to the quadratic equation for \( t^* \) is given by (14).

Having established the time it takes to rendezvous with a UGV traveling in a straight line, we now extend it to the more realistic case of a UGV convoy traveling in a piecewise-linear path. In particular, assume that the convoy travels at constant speed \( V_g \) on a piecewise-linear 2D path given by \( Z \) waypoints: \( WP_1, \ldots, WP_Z \), where the distance between consecutive waypoints \( WP_i \) and \( WP_{i+1} \) is given by \( d_{i,i+1} > 0 \), for \( i = 1, \ldots, Z - 1 \). Suppose the UGV convoy starts at \( WP_1 \) at time 0. Let \( w(t) \), the index of the most recently traversed waypoint by the convoy at time \( t \), be given by

\[
w(t) = \arg \min_{i=1,\ldots,Z-1} V_g t < \sum_{k=1}^i d_{k,k+1}. \tag{18}\]

The parameterized position of the UGV convoy at time \( t \) can then be computed as

\[
x_g(t) = WP_{w(t)} + \frac{WP_{w(t)+1} - WP_{w(t)}}{d_{w(t),w(t)+1}} \left( V_g t - \sum_{k=1}^{w(t)-1} d_{k,k+1} \right). \tag{19}\]

Define the function \( \Theta \) such that \( \Theta(y_1,y_2,y_3) \) gives the angle formed by points \( y_1, y_2, y_3 \). The theorem below shows how to compute \( S_j(t) \), the time it takes for UAV \( j \) located at \( x_j(t) \) at time \( t \) to rendezvous with the UGV convoy whose position is given by \( x_g \).

**Theorem 2** Suppose UAV \( j \) is located at position \( x_j(t) \) at time \( t \) that can only travel in a straight line with ground track speed \( V_a \). \( S(x_j(t),t) \), the time it takes for the UAV to rendezvous with the UGV convoy whose path is parameterized by \( x_g \), can be computed using the following algorithm:

\[
S(x_j(t),t) \leftarrow \infty
\]

\[
T_{\text{elapsed}} \leftarrow 0
\]

\[
\text{for } k = w(t), \ldots, Z - 1 \text{ do}
\]

\[
T_{\text{rendezvous}} \leftarrow R(x_j(t), x_g(t + T_{\text{elapsed}}), \Theta(x_j(t), x_g(t + T_{\text{elapsed}}), WP_{k+1}), T_{\text{elapsed}})
\]

\[
T_{\text{nextWP}} \leftarrow \frac{1}{V_g}||WP_{k+1} - x_g(t + T_{\text{elapsed}})||
\]

\[
\text{if } T_{\text{rendezvous}} \text{ exists and } T_{\text{rendezvous}} \leq T_{\text{nextWP}} \text{ then}
\]

\[
S(x_j(t),t) \leftarrow T_{\text{elapsed}} + T_{\text{rendezvous}}
\]

\[
\text{break}
\]

\[
\text{else}
\]

\[
T_{\text{elapsed}} \leftarrow T_{\text{elapsed}} + T_{\text{nextWP}}
\]

\[
\text{end if}
\]

\[
\text{end for}
\]

When appropriate, we will use the following shorthand notation: \( S_j(t) = S(x_j(t),t) \).

This result can then be easily extended to find \( T_i(t) \), the time it takes for UAV \( i \) to first fly in a straight line to visit a threat \( \tau \), and then change course and fly in a straight line to rendezvous with the UGV. The scenario is illustrated in Figure 5.
\( \dot{x}_i(t) \xrightarrow{V_a} x_i(t + t_1) = \tau \)

\[ ||\tau - x_i(t)|| \]

\( \xrightarrow{V_a} \)

\( X_g(t) \xrightarrow{V_g} X_g(t + t_1) \xrightarrow{\theta} X_g(t + t_1 + t_2) = X_i(t + t_1 + t_2) \)

**Figure 5.** Diagram used in Theorem 3 for computing the time it takes for a UAV to first visit a threat, and then rendezvous with the UGV convoy.

**Theorem 3** \( T_i(t) \), the time it takes for UAV \( i \) located at \( x_i(t) \) at time \( t \) to first fly in a straight line with ground track speed \( V_a \) to visit a threat located at \( \tau \), and then change course and fly straight to rendezvous with the UGV convoy whose path is parameterized by \( x_g \), is given by

\[
T_i(t) = \frac{||\tau - x_i(t)||}{V_a} + S(\tau, t + \frac{||\tau - x_i(t)||}{V_a}).
\]

**Proof 3** The first term in \( T_i(t) \) \((t_1 \text{ as shown in Figure 5})\) gives the time it takes for the UAV to fly in a straight line to visit the target at a ground track speed of \( V_a \). The second term \((t_2 \text{ as shown in Figure 5})\) makes use of the result in Theorem 2 to calculate the time it takes the UAV to rendezvous with the convoy starting from the target.

**IV. Experimental Validation**

To highlight the robustness of the proposed algorithm as well as to show that it can indeed be successfully deployed in an actual environment characterized by numerous computational and communications limitations, we implemented the algorithm on a testbed with both UAVs and UGVs.

**A. Hardware Platform**

The unmanned teams consist of a team of Khepera III Unmanned Ground Vehicles (UGVs) and a team of AR.Drone Unmanned Air Vehicles (UAVs). The choice of these vehicles is based on availability, cost, and satisfactory performance. Both these platforms benefit from ARM processors and an embedded Linux OS. AR.Drone is pre-packaged with a stabilizing autopilot. A laptop with limited computational power served as the pilot interface and drone server (ground station), where the drone server is responsible for providing high-level commands to the unmanned teams and also facilitating the pilot’s interface to the system. The A.R. Drone’s wireless communications are modified to work in managed mode (as opposed to ad-hoc mode), allowing all drones to connect to the same router (network). The drone server and the Khepera UGVs are also connected to the same network. Calculations show that the total required bandwidth is approximately 0.5% of the available communication bandwidth. A more detailed account of the bandwidth requirements and allocation is provided later in this paper. Figure 6 depicts the system level architecture of the proposed concept. A group of eight Vicon cameras are used for localization of the unmanned assets. The outputs of these cameras are interpreted by the Vicon server and the position and orientation data for all the vehicles
are broadcasted to the drone server on a separate network. We envision that this setup can be modified to receive GPS localization data directly to the UAVs and UGVs in an outdoor setting.

![System-level architecture: Team of UAVs and UGVs are connected through a WiFi router and receive high-level commands from a low power server/pilot interface.](image)

**Figure 6.** System-level architecture: Team of UAVs and UGVs are connected through a WiFi router and receive high-level commands from a low power server/pilot interface.

### B. Software Framework

Willow Garage’s Robotics Operating System (ROS) is chosen as the framework for the software development. ROS provides operating system-like functionality to our implemented algorithms, such as hardware abstractions and low-level drivers to interact with the hardware, as well as, message-passing between multiple processes. ROS is written in C++ and operating system agnostic allowing us to deploy it on any hardware designated as our ground station.

ROS’s capabilities are extended by writing low-level drivers for both the UGVs and UAVs. These low-level drivers allow us to send actuator controls to the vehicles and receive sensor information. In addition, interface drivers to the Vicon motion capture system are developed, such that a GPS-like hardware abstraction can be provided. These low-level drivers provide all of the functionality needed for the algorithms to interact with the hardware.

Each algorithm (or controller) is its own process that publishes and subscribes to a variety of information channels. ROS provides these information channels to allow processes to pass messages amongst each other as in Figure 7. For example, a higher level process may use an information channel to notify a process to change a parameter in response to an event in the system.

Specifically, the UGV convoy behaviors were implemented using three higher level processes: a spline generator, a waypoint supervisor, and a non-linear unicycle controller. The waypoint supervisor is provided with the current location of the convoy from the Vicon process (GPS) and a set of waypoints. It shares these waypoints with a second process, the spline generator, which returns a finer set of waypoints for the convoy to follow. The unicycle controller, a third process, drives the UGVs to the waypoints (while avoiding inter-vehicle collision) and receives a new waypoint.
from the waypoint supervisor each time a waypoint is cleared.

Similarly, the UAV behaviors were also implemented using a number of higher level processes, namely a PID (proportional-integral-derivative) controller, convoy protection, and threat assignment. The PID controller is responsible for driving the UAVs to any position. The convoy protection (which uses the PID controller) follows the UGV convoy using the convoy’s location, while also maintaining a spacing to avoid in-air collisions. The threat assignment process is responsible for detecting a threat and assigning one of the UAVs to investigate.

Figure 8 shows an overview of the ROS processes and the information passed among the processes in our implementation.

Figure 7. ROS Driver Architecture.

Figure 8. Process overview.
C. Limited Resource Operation

The assignment and control algorithms are implemented on a limited computational unit to demonstrate the low computation needs of the control architecture and algorithms.

To support high messaging frequencies, when possible, communication between Drone Server and each vehicle is implemented in UDP. The Effective Total Available Bandwidth takes into account inherent losses in communication through the 802.11g router and is estimated at 30.5 MBits/sec. Total bandwidth usage (inter-vehicle messages and command and localization messages) including bandwidth overhead for passing messages was measured empirically for the system at 146.744 KBits/sec.\(^5\)

D. Experimental Results

A hardware demonstration which serves as an experimental validation of the algorithms discussed in the previous sections was performed. It consists of a mission that incorporates all details highlighted in this paper. The mission is structured as follows:

1. The UGVs are given a set of waypoints which define a path to be taken through the (possibly hostile) environment. All UAVs are assigned to protect the convoy by flying over the convoy while maintaining some spacing from one another.

2. Once a possible target has been identified. A single UAV is assigned to fly over the threat, while the remaining UAVs remain over the convoy.

3. After the threat is neutralized (cleared), the assigned UAV returns to once again protect the convoy.

4. In the event that a threat cannot be neutralized (is persistent), the UAV signals the convoy to replan its path to avoid the threat.

The plots in Figure 9 show the recorded trajectories of the UAVs and convoy during the experiment. In particular, Figure 9(a) shows both UAVs performing convoy protection by flying above the UGVs as they follow their intended path. Notice that the UAVs maintain a safe separation from one another. A photograph of this operation is shown in Figure 10(a). Figure 9(b) shows the scenario when a non-persistent threat is encountered. In response, the UAVs execute the task allocation algorithm from Section 2 to determine which UAV gets dispatched to clear the threat so as to balance fuel consumption. Upon clearing the threat, the UAV returns to follow the convoy and the UGVs proceed in their originally intended path. A photograph showing a UAV examining the threat is shown in Figure 10(b).

Figure 9(c) shows the convoy encountering a persistent threat which blocks its intended path. Once again, the UAVs execute the task allocation algorithm and dispatches a UAV to visit the threat. When it is determined that the threat is persistent, the UAV informs the convoy that it must replan the path so as to avoid the threat. Afterwards, the UAV returns to the convoy and the UGVs proceed in following the recomputed path. To further illustrate the replanning of the path, Figure 11(a) shows the originally intended path of the convoy corresponding to the trajectories in Figure 9(a) and 9(b). Figure 11(b) shows the recomputed path taken by the convoy so as to avoid the persistent threat.

To validate that the task allocation algorithm in Section 2 performs as expected, a separate experiment was conducted in which 16 threats were presented to the convoy over an extended period of time. Figure 12 shows the fuel
Figure 9. Plots showing the trajectories of UAVs and the convoy as both a non-persistent and persistent threat are encountered.

(a) Two UAVs perform convoy protection on UGVs.
(b) A UAV is dispatched to clear a non-persistent threat.
(c) A UAV identifies a persistent threat and the convoy replans its path.

Figure 10. Photos showing the convoy protection and threat neutralization as carried out by the hardware platform.

(a) Two UAVs protect the convoy while maintaining spacing.
(b) One UAV is dispatched to visit a threat.
Figure 11. Plots showing the difference in path taken by the ground convoy before and after being informed about the presence of a persistent threat.

Figure 12. Plot showing the fuel consumption of the two UAVs while performing convoy protection.

V. Conclusions

Coordination of heterogeneous unmanned teams in resource constraint environments is of paramount importance in successful deployment of a number of unmanned missions. In this paper, we developed an assignment algorithm suitable for deployment in resource constraint environments and put that to the test on a hardware platform designed for this purpose. Although Multi-UAV convoy protection scenario has been tested as a potential application of this
algorithm and testbed, the testbed is designed such that it can be used for rapid small scale testing of coordination algorithms in resource limited environment. The software architecture used also allows for easy addition of different platforms to the existing system, rendering the platform even more versatile.

References