Real-time Multi-UAV Task Assignment in Dynamic and Uncertain Environments[*]

Luca F. Bertuccelli [†] Han-Lim Choi [‡] Peter Cho [§] and Jonathan P. How [¶]

Massachusetts Institute of Technology

This paper analyzes task assignment for heterogeneous air vehicles using a guaranteed conflict-free assignment algorithm, the Consensus Based Bundle Algorithm (CBBA). We extend this recently proposed algorithm to handle two realistic multi-UAV operational complications. Our first extension accounts for obstacle regions in order to generate collision free paths for UAVs. Our second extension reduces task planner sensitivity to sensor measurement noise, and thereby minimizes churning behavior in flight paths. After integrating our enhanced CBBA module with a 3D visualization and interaction software tool, we simulate multiple aircraft servicing stationary and moving ground targets. Preliminary simulation results establish that consistent, conflict-free multi-UAV path assignments can be calculated on the order of a few seconds. The enhanced CBBA consequently demonstrates significant potential for real-time performance in stressing environments.

I. Introduction

The military need for Unmanned Aerial Vehicles (UAVs) in theaters around the world has significantly grown over the past decade. The increasing demand has brought into focus several challenges associated with multiple UAV operation. In particular, the military has identified reducing UAV dependence on limited numbers of expert human pilots as a pressing concern. Currently, UAVs such as Predators require the full attention of two human operators. With this men-to-machine ratio, there will simply not be enough trained pilots to fly all the Predator missions expected in the future. Moreover, each Predator team of men and machine acts independently at present. In order to successfully carry out complex multi-platform missions, these teams will have to share information and cooperate among each other to perform in an optimal manner. Finally, since the battlespaces in which UAVs fly are highly dynamic, unmanned aircraft will need to react quickly to sudden changes in the environment, both on the ground and in the air.

All of these real-world challenges motivate serious investigation of automatic multi-UAV command and control[1–4]. While autonomous vehicles are unlikely to ever exhibit the same degree of judgment as expert pilots, they can help solve navigation and targeting problems which are too complicated for humans to tackle in real time. Autonomous systems can also deal with uncertainty about ground environments whose true states can only be indirectly inferred from noisy observations. In the presence of inevitable errors, a hierarchy of algorithms, ranging from low-level path

[*]This work was sponsored by the Department of the Air Force under Air Force Contract No. FA8721-05-C-0002. Opinions, interpretations, conclusions and recommendations are those of the authors and are not necessarily endorsed by the United States Government.

[†]Postdoctoral Associate, Dept of Aeronautics and Astronautics, lucab@mit.edu

[‡]Postdoctoral Associate, Dept of Aeronautics and Astronautics, hanlimc@mit.edu

[§]Research Staff, Lincoln Laboratory, cho@ll.mit.edu

[¶]Professor, Dept. of Aeronautics and Astronautics, Associate Fellow AIAA, jhow@mit.edu planning to high-level task allocation, will be needed to intelligently assign different vehicles to cover different regions of battle environments.

The speed with which future autonomous systems will assist with live battle decision making represents another critically important operational issue. Real-time task assignment algorithms originally conceived in the Operations Research (OR) community are being extended for applications in the UAV community[5–7]. While problems involving small numbers of autonomous agents can typically be solved sufficiently quickly for real-time implementation, the combinatorial complexity of task assignment becomes evident as the numbers of agents grows. Recent developments in task assignment[6,8–13] have focused on reducing computation times for these challenging problems.

Multi-UAV command and control is further complicated by its inherently distributed nature which strongly influences the choice of network architecture for decision making among aircraft. Centralized architectures can resolve conflicting situational awareness data collected by independent sensors at different battlespace vantage points via a single fusion protocol; however, they are often not operationally feasible due to significant communication overheads and over-reliance on a central decision-maker susceptible to failure. In contrast, decentralized decision-making architectures typically offer more robustness, but they are sensitive to information discrepancies across the UAV team. Furthermore, information agreement techniques that rely on implicit coordination and consensus[14,15] can be limiting, for the time required to reach consensus may be significant. The world may evolve so rapidly that the agreement becomes obsolete[6]. Defining a *sufficient* level of consensus remains an open question, as slight differences in information can lead to inconsistent and conflicting decisions.

Recently, the Consensus Based Bundle Algorithm (CBBA) has been proposed as a solution to several of these problems[8,9]. It can account for inconsistent information among the distributed agents, yet still output a conflict-free assignment for all the agents in the team. CBBA utilizes a market-based decision strategy for decentralized task selection, and it employs a consensus routine based on local communication to resolve conflicts and achieve agreement on winning bid values. While CBBA theoretically guarantees 50% of optimal performance, it has been empirically shown to perform within 93% of the optimal solution[8,9].

Although CBBA ensures a conflict-free solution even with differing situational awareness, its original implementation suffers from two limitations. Firstly, CBBA does not explicitly account for obstacle avoidance. Secondly, CBBA assignments can be extremely sensitive to input noise and produce churning behavior[5,16]. In this paper, we remedy these shortcomings. We first extend the Consensus Based Bundle Algorithm to account for polygonal obstacles in the environment. As we shall demonstrate, our modification only slightly increases the computational burden compared to the obstacle-free case. We also introduce a churning-mitigation formulation to reduce the algorithm's sensitivity to sensing noise. We present results on the obstacle avoidance extension as well as some preliminary work (to be augmented in our paper's final version) on flight path churning mitigation.

Our paper is organized as follows. The Consensus Based Bundle Algorithm is first reviewed in Section II. We then discuss our theoretical enhancements of CBAA in Sections III and IV which handle obstacles and noise churning. In Section V, we integrate the improved algorithm with a 3D simulator and present preliminary results for multiple aircraft servicing stationary and mobile targets in a Baghdad-like environment. Finally, we conclude in Section VI with some thoughts on future work.

II. Decentralized Task Assignment: Consensus-Based Bundle Algorithm

II.A. Problem definition

Given a list of N_t tasks and N_u agents, the goal of the task assignment is to find a conflict-free matching of tasks to agents that maximizes some global reward. An assignment is said to be free of conflicts if each task is assigned to no more than one agent. Each agent can be assigned a maximum of L_t tasks, and the assignment is said to be completed once $N_{\min} \triangleq \min\{N_t, N_u L_t\}$ tasks have been assigned. The global objective function is assumed to be a sum of local reward values, while each local reward is determined as a function of the tasks assigned to each agent.

The task assignment problem described above can be written as the following integer (possibly nonlinear) program with binary decision variables x_{ij} that indicate whether or not task j is assigned to agent i:

$$
\begin{aligned}
\max \quad & \sum_{i=1}^{N_u} \left(\sum_{j=1}^{N_t} c_{ij}(\mathbf{x}_i, \mathbf{p}_i) x_{ij} \right) \\
\text{subject to:} \quad & \sum_{j=1}^{N_t} x_{ij} \leq L_t, \ \forall i \in \mathcal{I} \\
& \sum_{i=1}^{N_u} x_{ij} \leq 1, \ \forall j \in \mathcal{J} \\
& \sum_{i=1}^{N_u} \sum_{j=1}^{N_t} x_{ij} = N_{\min} \triangleq \min\{N_t, N_u L_t\} \\
& x_{ij} \in \{0,1\}, \ \forall (i,j) \in \mathcal{I} \times \mathcal{J}
\end{aligned}
\tag{1}
$$

where $x_{ij} = 1$ if agent i is assigned to task j, and $\mathbf{x}_i \in \{0,1\}^{N_t}$ is a vector whose j-th element is x_{ij}. The index sets are defined as $\mathcal{I} \triangleq \{1, \ldots, N_u\}$ and $\mathcal{J} \triangleq \{1, \ldots, N_t\}$. The vector $\mathbf{p}_i \in (\mathcal{J} \cup \{\emptyset\})^{L_t}$ represents an ordered sequence of tasks for agent i; its k-th element is $j \in \mathcal{J}$ if agent i conducts j at the k-th point along the path, and becomes $\emptyset$ (denoting an empty task), if agent i conducts less than k tasks. The summation term inside the parenthesis represents the local reward for agent i. The *score function* $c_{ij}(\mathbf{x}_i, \mathbf{p}_i)$ can be any nonnegative function of either assignment $\mathbf{x}_i$ or path $\mathbf{p}_i$ (usually not a function of both). In the context of task allocation for autonomous vehicles with mobility, the score function often represents a path-dependent reward such as path length, mission completion time, or time-discounted value of target.

II.B. Algorithm

The scoring function in (1) can depend on the assignment $\mathbf{x}_i$ or the path $\mathbf{p}_i$ for $L_t > 1$. To address this dependency, previous combinatorial auction methods[10–13] treated each assignment combination (bundle) as a single item for bidding which led to complicated winner selection methods. In this section we review the Consensus Based Bundle Algorithm[8,9]. In CBBA, each agent has a list of tasks potentially assigned to itself, but the auction process is done at the task level rather than at the bundle level. CBBA consists of iterations between two phases: a bundle construction and conflict resolution.

II.B.1. Phase 1: Bundle Construction

The first phase of CBBA is bundle construction. In contrast to the bundle algorithms in [10–13] which enumerate all possible bundles for bidding, each CBBA agent creates just a single bundle and updates it as the assignment process progresses. During phase 1 of the algorithm, each agent continuously adds tasks to its bundle until it is incapable of adding any others.

Each agent carries four vectors: a winning bid list $\mathbf{y}_i \in \mathbb{R}_+^{N_t}$, a winning agent list $\mathbf{z}_i \in \mathcal{I}^{N_t}$, a bundle $\mathbf{b}_i \in (\mathcal{J} \cup \{\emptyset\})^{L_t}$, and the corresponding path $\mathbf{p}_i \in (\mathcal{J} \cup \{\emptyset\})^{L_t}$. Tasks in the bundle are ordered based on which ones were added first in time, while those in the path are ordered based on their assignment location. Note that the cardinality of $\mathbf{b}_i$ and $\mathbf{p}_i$ cannot be greater than the maximum assignment size L_t. Let $S_i^{\mathbf{p}_i}$ be defined as the total reward value for agent i performing the tasks along the path $\mathbf{p}_i$. In CBBA, if a task j is added to the bundle $\mathbf{b}_i$, it incurs the marginal score improvement

$$c_{ij}[\mathbf{b}_i] = \begin{cases} 0, & \text{if } j \in \mathbf{b}_i \\ S_i^{\mathbf{p}_i}, & \text{otherwise} \end{cases} \tag{2}$$

where $|\cdot|$ denotes the cardinality of the list, and $\oplus_n$ denotes the operation that inserts the second list right after the n-th element of the first list. [a] In other words, the CBBA scoring scheme inserts a new task to the location that incurs the largest score improvement, and this value becomes the marginal score associated with this task given the current path. Thus, if the task is already included in the path, it does not provide any additional improvement in score. Also, it is assumed that the addition of any new task provides nontrivial reward; namely, $c_{ij}[\mathbf{b}_i] \geq 0$ and the equality holds only when $j \in \mathbf{b}_i$.

The score function is initialized as $S_i^{\{\emptyset\}} = 0$, while the path and bundle is recursively updated as

$$\mathbf{b}_i = \mathbf{b}_i \oplus_{\text{end}} \{J_i\}, \;\; \mathbf{p}_i = \mathbf{p}_i \oplus_{n_{i,J_i}} \{J_i\} \tag{3}$$

with $J_i = \arg\max_j (c_{ij}[\mathbf{b}_i] \times h_{ij})$, $n_{i,J_i} = \arg\max_n S_i^{\mathbf{p}_i \oplus_n \{J_i\}}$, and $h_{ij} = \mathbb{I}(c_{ij} > y_{ij})$ where $\mathbb{I}(\cdot)$ denotes the indicator function that equals unity if the argument is true and zero if it is false. The recursion continues until either $|\mathbf{b}_i| = L_t$ or $\mathbf{h}_i = \mathbf{0}$. Notice that with (3), a path is uniquely defined for a given bundle, while multiple bundles might result in the same path.

II.B.2. Phase 2: Conflict resolution

CBBA agents add tasks to their bundle based on their currently assigned task set. Suppose that an agent is outbid for a task and thus releases it. The marginal score values for the tasks added to the bundle after this task are then no longer valid. The agent therefore also needs to release all tasks added after the outbid task. Otherwise, the agent would make further decisions based on wrong score values and thereby possibly degrade performance.

In the conflict resolution phase, three vectors are communicated for consensus. Two were described in the bundle construction phase: the winning bids list $\mathbf{y}_i \in \mathbb{R}^{N_t}$ and the winning agent list $\mathbf{z}_i \in \mathcal{I}^{N_t}$. The third vector $\mathbf{s}_i \in \mathbb{R}^{N_u}$ represents the time stamp of the last information update from each of the other agents. Whenever a message is passed, the time vector is populated with

$$s_{ik} = \begin{cases} \tau_r, & \text{if } g_{ik} = 1 \\ \max_{m:g_{im}=1} s_{mk}, & \text{otherwise} \end{cases} \tag{4}$$

where τ_r is the message reception time.

When agent i receives a message from agent k, $\mathbf{z}_i$ and $\mathbf{s}_i$ are used to determine which agent's information is the most up-to-date for each task. There are three possible actions agent i can take on task j:

[a] In later parts of this paper, the notion of $\oplus_{\text{end}}$ will also be used to denote the operation to add the second list at the end of the first one.

1. *update*: $y_{ij} = y_{kj},\ z_{ij} = z_{kj}$
2. *reset*: $y_{ij} = 0,\ z_{ij} = \emptyset$
3. *leave*: $y_{ij} = y_{ij},\ z_{ij} = z_{ij}$.

Table 1 in[8,9] outlines the decision rules. The first two columns of the table indicate the agent that each of the sender k and receiver i believes to be the current winner for a given task; the third column indicates the action the receiver should take, where the default action is *leave*.

If a bid is changed as an outcome of communication, each agent checks if any of the updated or reset tasks were in their bundle. If so, those tasks, along with all others added to the bundle after them, are released:

$$\begin{aligned} &y_{i,b_{in}} = 0,\ z_{i,b_{in}} = \emptyset,\ \forall n > \bar{n}_i \\ &b_{in} = \emptyset,\ n \geq \bar{n}_i \end{aligned} \tag{5}$$

where b_{in} denotes the n-th entry of bundle $\mathbf{b}_i$, and $\bar{n}_i = \min\{n : z_{i,b_{in}} \neq i\}$. It should be noted that the wining bid and the winning agent for the tasks added after $b_{i,\bar{n}_i}$ are reset, because removal of b_{in} can change scores for all the ensuing tasks. From here, the algorithm returns to the first phase and new tasks are added.

II.B.3. Time-discounted reward

It has previously been shown that if the scoring function satisfies a certain condition, called diminishing marginal gain (DMG), CBBA is guaranteed to create a conflict-free assignment [8,9] . For a DMG scoring function, the value of a task does not increase as other elements are added to the set before it. In other words,

$$c_{ij}[\mathbf{b}_i] \geq c_{ij}[\mathbf{b}_i \oplus_{\text{end}} \mathbf{b}] \tag{6}$$

for all $\mathbf{b}_i, \mathbf{b}, j$ such that $((\mathbf{b}_i \oplus_{\text{end}} \mathbf{b}) \oplus_{\text{end}} \{j\}) \in (\mathcal{J} \cup \{\emptyset\})^{L_t}$ where $\emptyset$ denotes an empty task. Since the marginal score of task j is defined as (2), the condition (6) can also be expressed in terms of the total score as

$$\begin{aligned} &\max_{n \leq |\mathbf{p}_i|} S_i^{\mathbf{p}_i \oplus_n \{j\}} - S_i^{\mathbf{p}_i} \\ &\quad \geq \max_{n \leq |\mathbf{p}_i|+1} \max_{m \leq |\mathbf{p}_i|} S_i^{(\mathbf{p}_i \oplus_m \{k\}) \oplus_n \{j\}} - \max_{m \leq |\mathbf{p}_i|} S_i^{\mathbf{p}_i \oplus_m \{k\}} \end{aligned} \tag{7}$$

for all $\mathbf{p}_i, j, k$ such that $((\mathbf{p}_i \oplus_m \{k\}) \oplus \{j\}) \in (\mathcal{J} \cup \{\emptyset\})^{L_t}$.

Note that many reward functions in search and exploration problems for autonomous agents are DMG. Consider specifically the following time-discounted reward[17–19]:

$$S_i^{\mathbf{p}_i} = \sum \lambda_j^{\tau_i^j(\mathbf{p}_i)} \bar{c}_j. \tag{8}$$

Here $\lambda_j < 1$ is the discounting factor for task j, $\tau_i^j(\mathbf{p}_i)$ is the estimated time agent i will take to arrive at task location j along the path $\mathbf{p}_i$, and $\bar{c}_j$ is the static score associated with performing task j. The time-discounted reward can model search scenarios in which uncertainty growth with time causes degradation of the expected reward for visiting a certain location. Eqn 8 also models planning of service routes in which client satisfaction diminishes with time. Since the triangular inequality holds for the actual distance between task locations,

$$\tau_i^j(\mathbf{p}_i \oplus_n \{k\}) \geq \tau_i^j(\mathbf{p}_i),\ \forall n, \forall k. \tag{9}$$

In other words, if an agent moves along a longer path, it arrives at each of the task locations at a later time than if it moves along a shorter path, resulting in further discounted score value. So for all nonnegative constants $\bar{c}_j$'s, $S_i^{\mathbf{p}_i}$ in (8) is DMG.

III. Extensions to CBBA: Obstacle Avoidance

CBBA represents a promising approach to the task assignment problem for multiple heterogeneous vehicles. However, in its original formulation, CBBA did not handle obstacle avoidance. Keep Out Zones arising from Surface-to-Air Missile sites or weather hazards represent real-world complications which UAV systems will generally confront. Hence, collision avoidance constraints are incorporated into CBBA in this section.

III.A. Collision avoidance

There are various approaches for incorporating obstacles into the CBBA. One intuitive strategy accounts for obstacles at the bundle construction stage by calculating new target scores. This modification requires that the tasking algorithm find overall distances around obstacles. Rather than using the minimum time (straight-line path) in the score calculation, the new score is instead calculated using the nominal path $\mathbf{p}_i$ perturbed by the distance δ_p required to fly around the obstacle for each vehicle. Hence the time-discounted scores becomes

$$S_i^{\tilde{\mathbf{p}}_i} = \sum \lambda_j^{\tau_i^j(\tilde{\mathbf{p}}_i)} \bar{c}_j \tag{10}$$

where $\tilde{\mathbf{p}}_i = \mathbf{p}_i + \delta_p$.

Unfortunately, this approach suffers from the fact that as tasks are added, and more importantly *removed*, during the conflict resolution stage of CBBA (Phase II), the computation required to find the collision-free path is effectively wasted, for that particular set of tasks is not added to the UAV's list. Furthermore, embedding the collision avoidance at the bundle selection level can significantly increase the iteration time during the conflict resolution stage, as the vehicles negotiate tasks among themselves.

We pursue an alternative obstacle avoidance approach in this paper. We first check the final CBBA assignments for collisions, then maneuver the vehicles around the obstacle regions, and finally apply corrections locally (in task list space) by inserting obstacle-free intermediate waypoints after solving a small shortest path algorithm. This procedure is summarized in Algorithm 1. The key difference with the preceding approach is employing a Dijkstra's algorithm solution for the shortest path.

A tacit assumption made by this algorithm is that obstacle dimensions are small relative to vehicle path lengths. This assumption holds for geographically dispersed waypoints, but we have seen additional benefit in this formulation for extremely dense environments as well.

III.B. Numerical results

We evaluated our obstacle avoidance CBBA modification via simulations where UAV number, target locations and obstacle field were all varied. The simulations were performed on a 150 $\times$ 250 lattice with initial UAV positions perturbed by a zero-mean Gaussian distribution with a variance of 50. We let our rectangular Keep Out Zones range in size from 10 $\times$ 20 to 80 $\times$ 20.

Table 1 displays mean times for running 20 Monte Carlo simulations on a 1.2GHz machine with 1GB of RAM. Even for a scenario with 7 UAVs, 7 targets and 3 obstacles, the computation time was still well under 1 second. The largest scenario tested at the time of this draft included 7 UAVs, 16 targets and 8 static obstacles. The mean run time for this case was found to be 2.37 seconds, with a maximum time of 4.67 seconds, and a minimum time of 0.86 seconds.

Algorithm 1 Obstacle avoidance in CBBA

```
Initialize target and UAV locations
Run CBBA algorithm for find optimal plan $\mathbf{p}^i \quad \forall i = 1, \ldots, N_u$
for all Vehicles $i$ do
  for all Tasks in bundle $j$ do
    Check for collision avoidance
    if Path traverses obstacles then
      Construct visibility graph $G$ from current waypoint $W_j$ to next waypoint $W_{j+1}$
      Run Dijkstra's algorithm and find shortest path $q_j^i$ to next waypoint, $q_j^i = \text{Dijkstra}(G)$
      Augment agent path with new path segment $\mathbf{p}_{1:j}^i \leftarrow \mathbf{p}_{1:j}^i \oplus_j q_j^i$
      Continue to next task in bundle
    end if
  end for
  Broadcast new path $\mathbf{p}^i$ to each UAV
end for
```

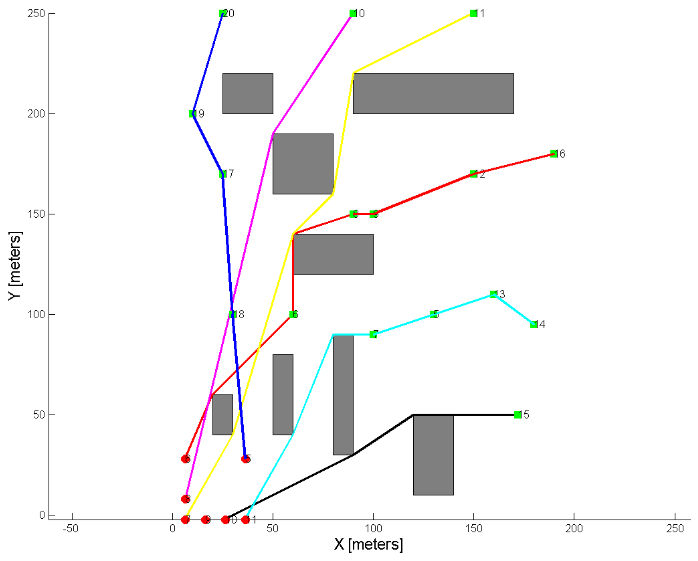


Figure 1: 7 UAV task assignment for 16 targets and 8 obstacles solved in approximately 2 seconds

IV. Targeting with Sensing Noise

Realistic multi-UAV mission planning must generally take into account not only ground obstacles but also situational awareness uncertainty[7,20]. Errors can arise from noisy sensors (such as cameras detecting non-existent targets or positioning inaccuracies) and from adversarial strategies designed to intentionally create ambiguity in the state of the world. If situational awareness errors are sufficiently large, they can force UAV control systems to unnecessarily replan missions already underway. Such replanning can result in vehicle flight path churning[5,16] wherein UAV assignments constantly flip and, in the extreme case, never converge. While churning problems can arise due to sensing noise in the location, identity, and motion of a target, this section considers the problem when sensing noise impacts the identity[7,21] of the target, and therefore has a direct impact on the target's score.

Table 1: Mean task assignment computation time $\overline{T}$ and standard deviation σ_T for N_u UAVs with N_t ground targets and 3 obstacles, on a 1.2GHz machine with 1GB of RAM

N_u	N_t	$\overline{T}$[seconds]	σ_T[seconds]
3	3	0.82	0.11
	5	0.74	0.11
	7	0.78	0.10
5	3	0.79	0.10
	5	0.75	0.08
	7	0.84	0.08
7	3	0.80	0.10
	5	0.75	0.11
	7	0.86	0.08

IV.A. Churning

The most straightforward technique to handling changes in situational awareness is to immediately react to new information by reassigning ground targets. In a deterministic sense, replanning proves to be beneficial since the parameters in the optimization are perfectly known. However in a stochastic setting, replanning may not be beneficial. For example, since the observations are corrupted by sensor noise, the key issue is that replanning *immediately* to this new information results in a task assignment control process with short dwell times (analogous to having a high bandwidth controller) that simply tracks the sensor noise. From the perspective of a human operator, continuous reassignments of the vehicles in the fleet may also prove to be undesirable (and lead to increased human errors), especially if this effect is due primarily to sensing errors.

A simple example of churning is shown in Figure 2, where two vehicles are assigned to visit the targets with the highest value. The target scores are each corrupted by additive Gaussian noise, $\tilde{c} = c_i + v_i$, where $v_i \sim N(0, R)$ is distributed according to a zero-mean Gaussian distribution with known covariance R. The original assignment of the vehicles (starting on the left) is to visit the bottom right target. At the next time step, the assignment for the vehicle is switched to the top right target due to simulated sensing noise. After the vehicles change direction towards that target, the assignment switches once again. As the mission progresses, the vehicle repeatedly alternates heading towards the two different targets. It converges to a fixed assignment only near the end of its mission.

Algorithm 2 Mitigating sensor noise in CBBA

```
Initialize target and UAV locations
Run CBBA algorithm for find optimal plan $\mathbf{p}^i \ \forall i = 1, \dots, N_u$
Obtain new (noisy) measurements on the targets score $\tilde{c}_{ij} = c_{ij} + v_{ij}$
Update target score values $\tilde{c}_{ij} := \begin{cases} \tilde{c}_{ij} + \Delta_{ij} & \text{If the previous bundle contained target } j \text{ for UAV i} \\ \tilde{c}_{ij} & \text{Else} \end{cases}$
Return to next assignment
```

IV.B. Churning mitigation

In previous work[5,22] we had proposed a filter-embedded approach (FETA) to ameliorate the churning phenomenon. The approach was to rewrite the optimization problem in Eq. 1 to incorporate

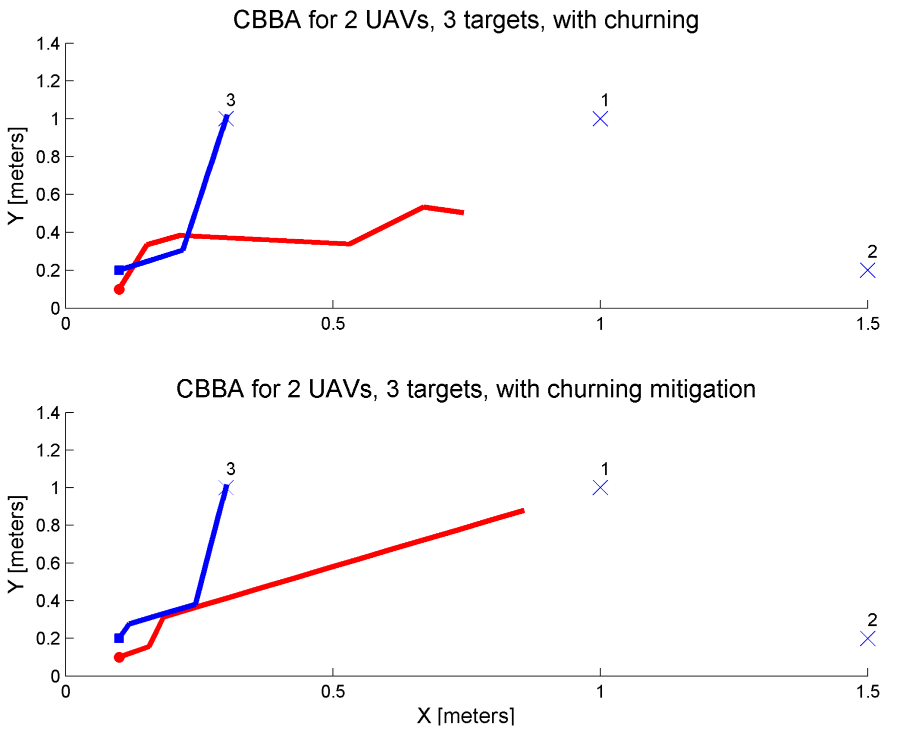


Figure 2: Example of churning in a 2 UAV, 3 target scenario with additive sensor noise. In the top plot (no churning mitigation), UAV 2 (red) continually switches between targets 1, 2, and 3 before finally settling upon target 2. (The optimal solution was target 1). The bottom plot displays results from Algorithm 3 which mitigates sensing noise. The UAV correctly settles upon target 1.

a penalty term for assignment changes into the objective function. While the original FETA formulation proposed to penalize the changes in the assignment, this work proposes to add a bonus term if the same task is assigned to the same agent:

$$
\begin{aligned}
\max \quad & \sum_{i=1}^{N_u}\left(\sum_{j=1}^{N_t}(c_{ij}(\mathbf{x}_i,\mathbf{p}_i)+\Delta_{ij}x_{ij}^o)x_{ij}\right)\\
\text{subject to:} \quad & \sum_{j=1}^{N_t} x_{ij} \leq L_t,\ \forall i \in \mathcal{I}\\
& \sum_{i=1}^{N_u} x_{ij} \leq 1,\ \forall j \in \mathcal{J}\\
& \sum_{i=1}^{N_u}\sum_{j=1}^{N_t} x_{ij} = N_{\min} \triangleq \min\{N_t, N_u L_t\}\\
& x_{ij} \in \{0,1\},\ \forall (i,j) \in \mathcal{I}\times\mathcal{J}
\end{aligned}
\tag{11}
$$

where x_{ij}^o is the incumbent solution. Namely, CBBA scoring is modified so that if a UAV contains a set of targets in its bundle, the UAV will increase the target scores by a modified cost $\tilde{c}_{ij} = c_{ij}+\Delta_{ij}$. If the UAV does not contain a target in its bundle, the target's score is unaffected. Although equivalent to the original FETA formulation in a subtractive form, this additive formulation facilitates incorporation of FETA concept into the CBBA framework. Note that the only needed change in the algorithm is slight modification of scoring schemes; this modification has no effect on the DMG

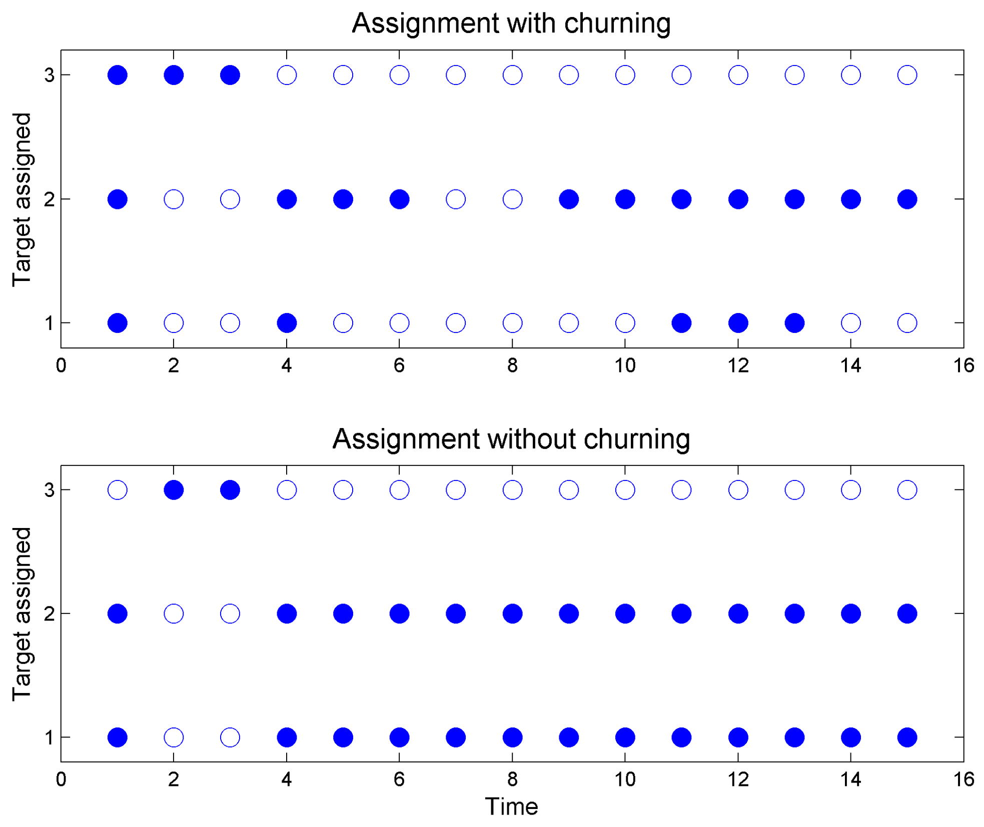


Figure 3: Visualization of assignment for the case of churning (top) and mitigated churning (bottom). Filled circles (empty circles) indicate the assignment for UAV 1 (2, respectively). The assignment switches frequently due to sensing noise (top plot), but settles down when churning is mitigated (bottom plot).

property of the scoring schemes and therefore on the algorithm convergence.

While Δ_{ij} can be interpreted as a tuning parameter introduced by the user, our current work is using the changes in the plan (as was done in FETA[5]) to adaptively determine Δ_{ij}.

IV.C. Numerical results

We simulated a 2 UAV, 3 target assignment scenario with sensing noise. This small example was chosen yield clear and intuitive visualizations. The effect of adopting the modified cost function can be seen in revisiting the churning example of Figure 2 (bottom). The blue UAV's flight path churns only slightly at the simulation's beginning. But both UAVs' plans quickly converge to churn-free assignments.

Figure 3 illustrates another example of assignment change. The top plot displays the effect of real-time replanning in the presence of noise; assignment oscillation is clearly visible. As the UAVs start the mission, UAV 1 is initially assigned to visit all the targets, while at the second and third time steps, it is assigned to visit only target 3. By time step 4, UAV 1 has been retasked to visit targets 1 and 2. Toward the end of the mission, UAV 1 is tasked to visit target 2, while UAV 2 is directed towards targets 1 and 3. The bottom plot in the figure illustrates the impact of churning mitigation. After a few initial transient switches, the assignment settles by time step 4 to UAV 1 visiting targets 2 and 3, and UAV 2 visiting target 1. This division of labor actually represents the optimal solution.

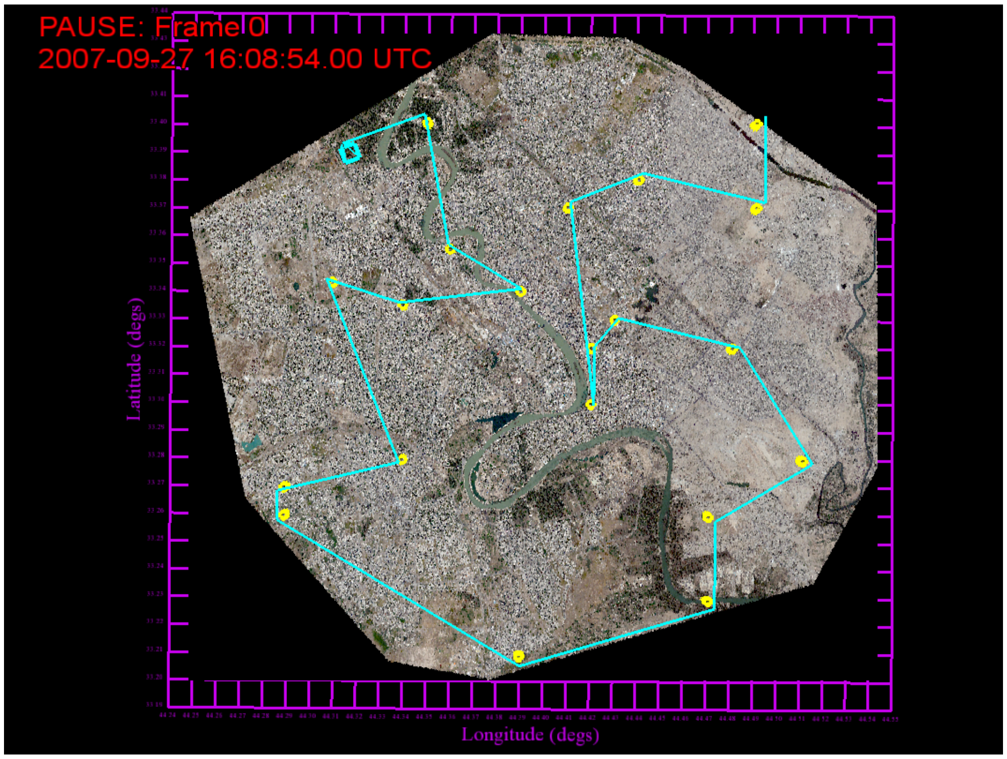


Figure 4: Twenty ground targets randomly chosen throughout a Baghdad-like environment colored in yellow. The cyan polyline represents the flight path calculated by CBBA for the UAV starting near the image's upper left corner.

V. Multi-UAV Command and Control Simulation

V.A. 3D visualization and interaction tool

In order to more rigorously test the efficacy of the Consensus Based Bundle Algorithm, we have integrated it into a prototype command and control environment which allows a human operator to intuitively interact with multiple aircraft. Our simulator provides a 4D visualization of the air and ground pictures. Its graphics front-end is based upon an open source, high performance 3D toolkit called OpenSceneGraph[b]. Our particular multi-UAV implementation has been adapted from previous simulation projects which were developed to handle large volumes of laser radar, video, and digital photo data sets[23]. Though our graphics tool does not enjoy the widespread popularity of other 3D visualization environments such as Google Earth[c], it can be customized for rapid human interaction with multiple UAVs in ways that Google Earth cannot. So we have found it useful for prototype development.

We choose to simulate one or more Predators flying over a Baghdad-like environment to model UAV missions of current operational interest. We therefore import into our tool commercial satellite imagery of Baghdad with 1 meter Ground Sampling Distance. Users can manipulate and view the Baghdad-like imagery at different levels of detail via mouse controls which intentionally mimic those of Google Earth: the left button translates, the center button rotates and the right button zooms.

Human operators can also interact with our simulator via specialized buttons designed specifically for multi-UAV command and control. Other menu buttons allow for target waypoints and

[b]http://www.openscenegraph.org/projects/osg
[c]http://earth.google.com/

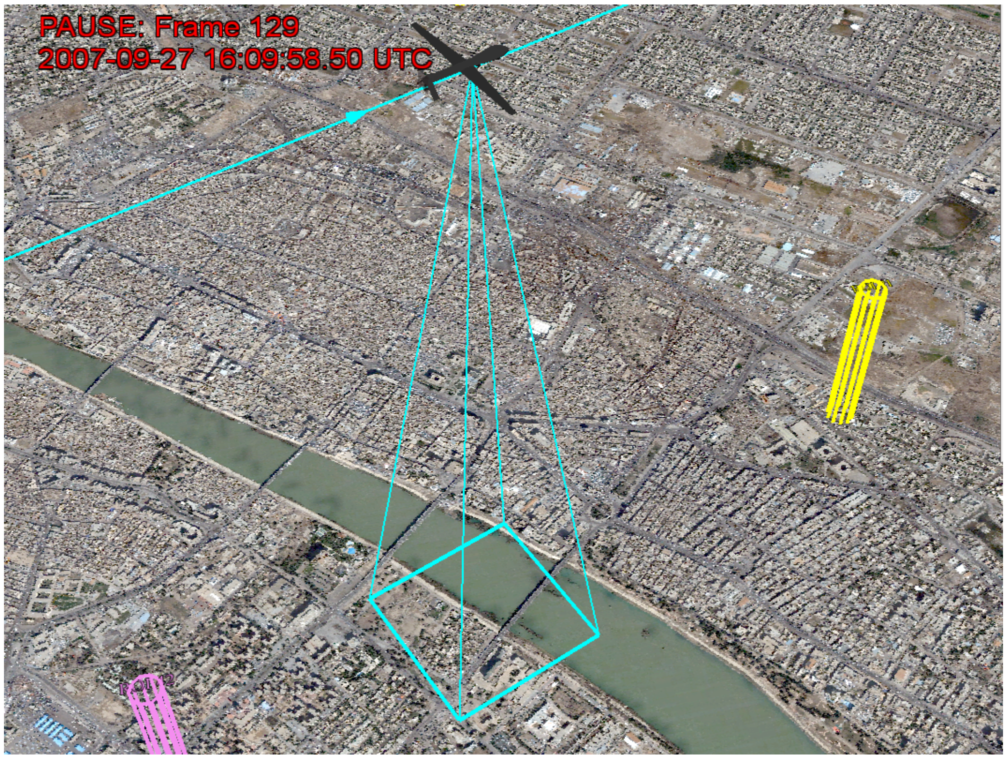


Figure 5: Closeup view of a UAV model with its observation frustum flying over a Baghdad-like environment. The pink coloring of the ground target near the image's bottom left corner indicates that it has been previously overflown.

Keep Out Zones to be manually entered. [d] Twenty such ground waypoints which were randomly selected throughout the Baghdad map are colored yellow in Figure 4.

A human operator may choose to manually enter a flight path passing through the waypoints as a polyline terminated via a mouse doubleclick. Or he may instead instruct the automated system to compute a flight path using CBBA by clicking on a menu button. A string message containing ground and air statevector information is then sent from the main C++ visualization program to an independent set of MATLAB codes via an ActiveMQ message queue [e]. In just a few seconds, the aircraft route is computed, returned and displayed as in Figure 4.

Our simulator comes with a clock whose time and state are listed in the viewport's upper left corner. Once the clock is set in motion, UAVs fly along their calculated paths at 36 meters/sec which corresponds to a typical Predator loiter speed. We also fix the first aircraft model's altitude at a constant value of 2.5 km, the second at 2.6 km, the third at 2.7 km, etc so that they never crash in three dimensions. For reasonable numbers of UAVs over our Baghdad-like environment, they all fly well below the maximum Predator ceiling of 25,000 ft.

As can be seen in the closeup view of Figure 5, our UAV models are endowed with observation frusta which indicate instantaneous fields-of-view for notional onboard video cameras. Frusta orientations relative to their aircraft platforms are specified by 3 angular degrees of freedom; currently, all cameras are restricted to nadir views. When a frustum intercepts a ground waypoint, the target is regarded as having been serviced, and its color is changed from yellow to pink. In any subsequent calls to the UAV path planner, previously serviced targets are ignored. Each ground target is consequently visited only once by one airplane in our simulations.

[d] These ground objects can also be read in automatically from a database.
[e] http://activemq.apache.org/

Once the simulator's clock has begun and a mission is underway, additional UAVs can be introduced to emulate the entrance of new aircraft into the battlespace. The user may then request that updated flight paths for all aircraft be replanned based upon the ground targets not yet overflown. The user can also select one or more flight paths and mark them for deletion. This removal capability models a shoot-down or malfunction of a UAV which is forced to withdraw. We can similarly simulate the response of a UAV fleet to ground targets that pop-up while a mission is underway.

The time required for CBBA to calculate flight paths is a function of the number of aircraft and ground targets. It of course also depends upon computer hardware. In Table 2, we list timing results for 3 trial sets of one through eight UAVs randomly started around the Baghdad airspace. The number of ground targets in each of these trials was fixed at 50. As the table results indicate, our simulation system's timing depends only weakly upon UAV number. In all trials, it returned results in less than 3 seconds. Our prototype's responsiveness is sufficiently fast to enable real-time reaction in future multiple-UAV command and control systems.

V.B. Dynamic ground movers

In urban combat situations, the ground targets of greatest intelligence value are often mobile. Full motion video monitoring of vehicle activity has become especially valuable for uncovering terrorist networks hidden within complex cities. Consequently, Predators on patrol often need to follow dynamic movers. In this subsection, we present preliminary simulation results of multiple UAVs responding to continuous changes on the ground.

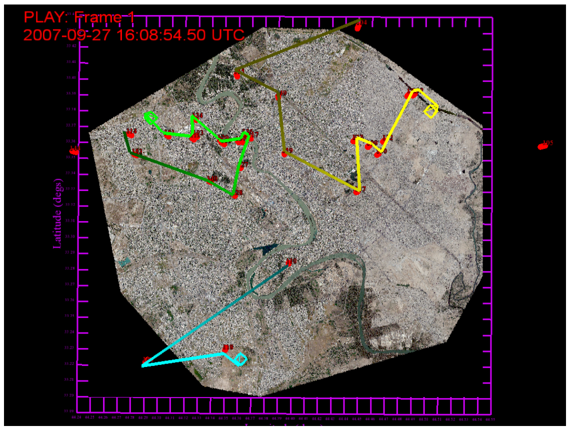
We start with GPS truth data collected during a Lincoln Laboratory field exercise in 2007 by 40 cars driving around the town of Lubbock, TX. The vehicles in this urban campaign performed typical driving maneuvers such as turning at corners, stopping at intersections and parking outside buildings. We translate and rescale the vehicle truth data so that they cover our Baghdad map and yet still maintain an average speed of 35 mph. The modified GPS tracks provide a challenging and realistic set of ground movers for our enhanced aerial path planning algorithm to chase.

We next initialize one or more UAVs within the Baghdad airspace and set the simulator's clock running. The main simulation program transmits state vectors for both the cars and aircraft to the CBBA Matlab module every 10 seconds in simulation time. The Matlab code asynchronously returns its path results sufficiently fast that the entire simulation runs smoothly in real time. Four representative snapshots illustrating 3 UAVs following the ground movers are displayed in Figure 6.

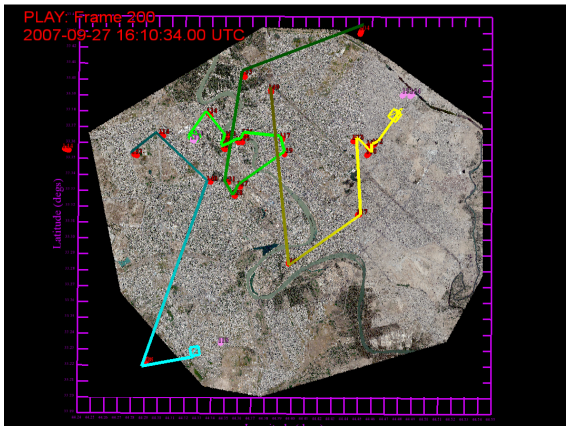
As the simulation proceeds, it ignores any vehicles which depart the battle space as defined by the purple grid in figure 6. On the other hand, a new vehicle entering the grid is added to the target list on the next flight path update request. As for our simpler static ground target simulations, a moving vehicle is regarded as having been serviced once it is overflown by some UAV. The car is then removed from the list of targets to be visited, and its color is changed from red to pink in the output display.

In these preliminary simulation runs, we assigned equal priority to each ground mover. The score function in Eq 8 consequently directs each airplane to pay more attention to nearby vehicles and less attention to distant targets. The end portions of flight paths located far away in space and time from a UAV's instantaneous position should therefore be regarded with less weight than the parts of the paths closer to the aircraft. It is important to note that the immediate flight plans for each of the UAVs pictured in Fig 6 do not significantly churn as the simulation runs even though the ground targets in their nearby vicinity constantly move in various directions.

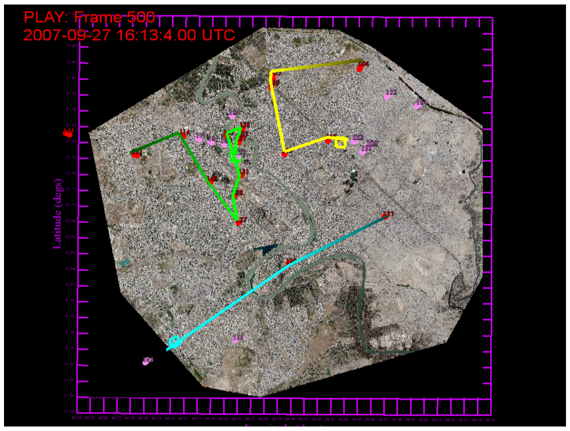
Several more realistic refinements remain to be added to our dynamic simulations such as nonzero aircraft turning radii, ground target location uncertainty and communication dropouts. Nevertheless, we believe these early results compellingly demonstrate the potential for automated



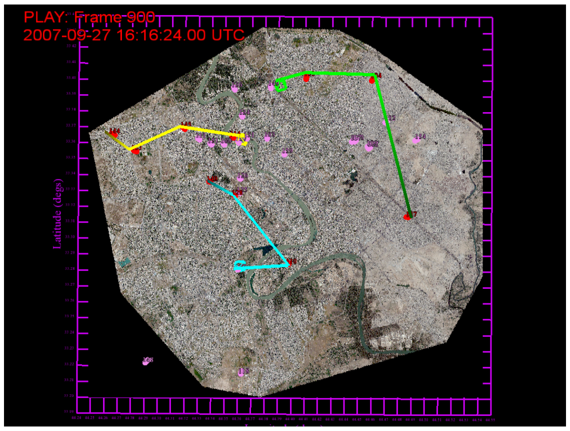
(a)



(b)



(c)



(d)

Figure 6: 4 snapshots from a movie of 3 Predators chasing vehicles moving with average 35 mph speeds. UAV flight paths are updated every 10 secs of simulation time. Time-decaying confidence in CBBA assignments for distant targets is heuristically indicated by flight path color fading. As the simulation proceeds, cars overflown by some UAV change color from red to pink. The first snapshot displays initial locations for cars and aircraft. The second, third and fourth snapshots illustrate the dynamic ground and air pictures at $\delta_t = 100$, 250 and 450 secs.

path planning tools to solve complex navigation and targeting problems in real time which are too complicated for human operators. As the number of UAVs executing stressing missions grows in the future, so will the need for such automated assistance.

VI. Conclusion and Ongoing Work

We have extended the Consensus Based Bundle Algorithm in two significant ways. Firstly, CBBA can now route UAVs safely past ground environment obstacles. Secondly, CBBA now mitigates churning of UAV flight paths induced by target situational awareness uncertainty. Obstacle avoidance incurs only a modest computation time increase relative to obstacle-free path planning, and it exhibits significant potential for future real-time task assigning. Furthermore, our churning mitigation approach yields vehicle assignments with decreased sensitivity to system noise. We have also integrated our enhanced algorithm into a simulation tool which permits intuitive human interaction with multiple air vehicles and ground targets.

Table 2: Simulation system time (in seconds) to plan and display flight paths for multiple UAVs servicing 50 static ground targets. The simulation trials were run on a Dell tower with 3 GHz Quadcore Xeon CPUs.

Number of UAVs	Trial 1	Trial 2	Trial 3
1	0.92	0.67	0.89
2	1.49	1.58	1.61
3	1.46	1.97	1.71
4	1.79	2.14	2.15
5	2.17	2.28	2.05
6	2.70	2.14	2.36
7	2.89	2.73	2.49
8	2.76	2.77	2.89

The results which we have presented in this paper raise many interesting questions which warrant future study. For example, we have assumed that at least approximate locations for all ground targets are known at the start of all UAV missions. In real battle conditions, it is much more likely that some targets will only be discovered once missions are underway. Other moving targets whose locations may be known at one time might well be later lost as they intentionally execute evasive maneuvers. Evolving ground target uncertainty over time via growing error ellipsoids represents one possible approach to methodically handling these seach and discovery problems.

We also look forward to incorporating kinematic and dynamic constraints on both air and ground movers into CBBA in the future. UAVs need to be penalized for attempting large heading changes. And vehicles usually drive along well-defined road networks which strongly constrain their likely future motions. We expect that extending the CBBA to incorporate these restrictions will render this algorithm of significant theoretical and tactical interest.

References

[1] Bellingham, J., Tillerson, M., Richards, A., and How, J. P., *Multi-Task Allocation and Path Planning for Cooperative UAVs.*, Kluwer Academic Publishers, 2003.

[2] Bethke, B., How, J., and Vian., J., “ Group Health Management of UAV Teams With Applications to Persistent Surveillance.” *IEEE American Controls Conference*, 2008.

[3] Bourgault, F., Furukawa, T., and Durrant-Whyte., H. F., “Decentralized Bayesian Negotiation for Cooperative Search.” *IEEE/RSJ International Conf. on Intelligent Robots and Systems*, 2004.

[4] Chinchuluun, A., Grundels, D., and Pardalos, P., “Searching for a Moving Target: Optimal Path Planning.” *IEEE International Conference on Networking, Sensing and Control*, 2005.

[5] Alighanbari, M., Bertuccelli, L. F., and How, J. P., “Filter-Embedded UAV Task Assignment Algorithms For Dynamic Environments,” *AIAA Conference on Guidance, Navigation and Control*, 2004.

[6] Alighanbari, M., *Robust and Decentralized Task Assignment Algorithms for UAVs*, Ph.D. thesis, MIT, 2007.

[7] Bertuccelli, L. F., Alighabari, M., and How, J. P., “Robust Planning For Coupled Cooperative UAV Missions.” *IEEE Conference on Decision and Control*, 2004.

[8] Brunet, L., Choi, H.-L., and How, J. P., “Consensus-based auction approaches for decentralized

task assignment," *AIAA Guidance, Navigation and Control Conference*, 2008.

9 Choi, H.-L., Brunet, L., and How, J. P., "Consensus-based decentralized auctions for robust task allocation," *IEEE Trans. on Robotics*, Vol. submitted, 2008.

10 Parkes, D. C. and Ungar, L. H., "Iterative Combinatorial Auctions:Theory and Practice," *Proceedings of the 17th National Conference on Artificial Intelligence*, 2000.

11 Berhault, M., Huang, H., Keskinocak, P., Koenig, S., Elmaghraby, W., Griffin, P., and Kleywegt, A., "Robot Exploration with Combinatorial Auctions," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003.

12 Andersson, A., Tenhunen, M., and Ygge, F., "Integer programming for combinatorial auction winner determination," *Proceedings. Fourth International Conference on MultiAgent Systems*, 2000.

13 de Vries, S. and Vohra, R., "Combinatorial auctions: A survey," *INFORMS Journal of Computing*, Vol. 15(3), 2003, pp. 284–309.

14 Ren, W., Beard, R. W., and Kingston, D. B., "Multi-agent Kalman consensus with relative uncertainty," *American Control Conference, 2005. Proceedings of the 2005*, 2005, pp. 1865–1870 vol. 3.

15 Ren, W., Beard, R. W., and Atkins, E. M., "A survey of consensus problems in multi-agent coordination," *American Control Conference, 2005. Proceedings of the 2005*, 2005, pp. 1859–1864 vol. 3.

16 Tierno, J. and Khalak, A., "Frequency domain control synthesis for time-critical planning," *Proceedings of the IEE European Control Conference,*, 2003.

17 Bellingham, J., Tillerson, M., Richards, A., and How, J., "Multi-Task Allocation and Path Planning for Cooperating UAVs," *Proceedings of Conference of Cooperative Control and Optimization*, Nov. 2001.

18 Alighanbari, M., *Task assignment algorithms for teams of UAVs in dynamic environments*, Master's thesis, Massachusetts Institute of Technology, 2004.

19 Alighanbari, M. and How, J. P., "Decentralized Task Assignment for Unmanned Aerial Vehicles," *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference*, 2005.

20 Bertuccelli, L. F., *Robust Planning for Heterogeneous UAVs in Uncertain Environments.*, Master's thesis, MIT, 2004.

21 Bertuccelli, L. F. and How, J. P., "Search for Dynamic Targets with Uncertain Probability Maps," 2006.

22 Alighanbari, M. and How, J. P., "A Robust Approach to the UAV Task Assignment Problem," *International Journal of Robust and Nonlinear Control*, Vol. 18, No. 2, 2008.

23 Cho, P., "3D Organization of 2D Urban Imagery," *SPIE Defense and Security Symposium (Orlando, FL)*, 2008.