Coordination and Communication of Cooperative Parafoils for Humanitarian Aid

Pini Gurfil[*] and Sharoni Feldman[†]
Faculty of Aerospace Engineering
Technion - Israel Institute of Technology, Haifa 32000, Israel

Moran Feldman[‡]
Computer Sciences Department, The Open University, Israel

Abstract

In case of a wide-scale disaster, an accurate airdrop of emergency supplies is crucial. In this paper, we present a novel top-down approach for designing and executing airdrop missions using guided parafoils. We develop a guidance algorithm and a cooperative task management method for autonomous handling of faults and exceptional events by the parafoil group. The autonomous operation is based on inter-parafoil ad-hoc communication. A parafoil or a number of parafoils can dynamically react to events that prevent one or more parafoils from successfully completing their mission. Two recovery methods are presented – Swap, which enables parafoils to dynamically exchange their targets, and Replace, which gives precedence to prioritized targets over low-priority targets. The parafoil guidance method, combined with the task management algorithm, significantly increases the probability of successful airdrop. The small overhead of the communication layer and the low complexity of the swap and replace recovery algorithms enable these procedures to run in a distributed environment under real-time limitations.

---

[*] Senior Lecturer. Email: pgurfil@technion.ac.il

[†] Research Associate. Email: sharoni@aerodyne.technion.ac.il

[‡] Graduate Student. Email: feldsh@netvision.net.il

1. Introduction

Multiagent robotics has seen significant progress in recent years. Studies were dedicated to developing a taxonomy for multiagent robotics [1], designing behavioral-based control using potential field theory [2], [3] and simulating flocking rules [4].

The ability of multiagent systems to meet complex mission requirements in an arbitrary theater under partial and uncertain information paved the way for applying multiagent methodologies to a variety of platforms in the air, ground and sea. Particular airborne platforms that have gained significant scholarly attention are Unmanned Aerial Vehicles (UAVs). Recent studies developed UAV cooperative control [5] and coordination [6] algorithms, while others have designed path planning [7] and task assignment [8] methods. While the literature on cooperative UAVs is abundant, the reported research efforts dedicated to developing cooperative *parafoils* are few [11].

Autonomous parafoils are equipped with sensors (typically GPS receivers and occasionally an inertial measurement unit) and actuators that enable autonomous operation and precision landing at designated sites (see e. g. the parafoil developed in the FastWing project [12]). This ability is critical in times of emergency such as natural disasters, as it ensures that urgent supplies will reach those in need of immediate support. Creating a flock of cooperative parafoils rather than a group of independent parafoils yields a more reliable, fail-safe system, in which one malfunctioning parafoil, pre-planned to deliver critical assistance (e.g. drinking water), may be replaced by another parafoil aimed at a target with lesser priority.

This paper presents a novel autonomous distributed task management method that enables parafoils to exchange landing sites among themselves in real time. As soon as a parafoil monitoring system predicts that the parafoil is unable to land at its pre-designated target, the monitoring system initiates a process for reassigning parafoils to targets. This

real-time distributed task management may result in a list of changes in existing pairings of parafoils and targets.

The distributed task management process utilizes a set of parameters such as the type of payload and the target priority in addition to the current position, velocity, gliding angle, wind speed and direction and other specific characteristics of each parafoil. The distributed task management process is applied autonomously by the parafoils without any external intervention, thus permitting stand-off release. The infrastructure for the target reallocation process comprises an ad-hoc communication protocol wherein every parafoil acts as a session end-point and as a relay that connects all parafoils as long as they are within transmission range.

An ad-hoc network is a communication network that enables communication among mobile wireless users without using a fixed set of base stations. Each user acts as a router (relay), allowing other users to communicate through its mobile communication device. The communication range of each device is limited; thus, at any given time a user can exchange packets only with other devices in its receiving/transmitting range. The set of users is highly dynamic: new users join in while other users may quit or move out of transmission range. In addition, each node can arbitrarily move and possibly cause loss of communication with some nodes while creating new connections with other nodes [9].

The main contribution of this work is the development of a novel method aimed at improving the chances of a successful parafoil airdrop. This method (i) renders the parafoils release procedure into a fully automatic and autonomous process that fuses individual parafoils and targets into a set of cooperative entities aimed at fulfilling a global mission; and (ii) increases the chances for successfully completing a single airdrop task by creating redundancy based on judicious task management without adding redundant parafoils.

2. Parafoil Dynamics, Trajectory Design and Guidance

In this section, we outline the flight mechanics of the parafoils and develop reference trajectories for each parafoil. The reference trajectory is tracked using neighboring optimal control.

2.1 Dynamical Model

The parafoil dynamical model is written in north-east-down (NED) coordinates, so that *x* and *y* are the north and east positions relative to the target, and *h* is the altitude above the surface. Under the assumption of equilibrium glide, the flight-path angle, γ, and the magnitude of the velocity, *V*, are constant. Denoting the wind velocity components in the NED coordinates by w_x and w_y (we assume that there is no wind shear), and the heading angle by ψ, the equations of motion assume the form [10]

$$\dot{x}(t) = V\cos\gamma\cos\psi(t) + w_x(t) \tag{1}$$

$$\dot{y}(t) = V\cos\gamma\sin\psi(t) + w_y(t) \tag{2}$$

$$\dot{h}(t) = V\sin\gamma \tag{3}$$

$$\dot{\psi}(t) = g\tan\phi(t)/V \tag{4}$$

In Eq. (4), ϕ denotes the bank angle, which, at steady state, assumes a constant value, ϕ_d, determined by a servo deflection, δ, so that the closed-loop bank angle dynamics are given by

$$\dot{\phi}(t) = [-\phi(t) + \phi_d(\delta)]/\tau \tag{5}$$

where τ is the equivalent time constant of the bank angle control loop.

The constant flight-path angle, γ, appearing in Eqs. (1)-(3), satisfies the relationship [13]

$$\tan\gamma = -\frac{C_D}{C_L \cos\phi_d} \tag{6}$$

where C_D and C_L are the drag and lift coefficient, respectively. Thus, the dynamical model (1)-(5) includes two control variables: γ and ϕ_d (the servo deflection enables control of the lift and drag coefficients). Hence, based on Eq. (4), at steady state,

$$\dot{\psi} = g \tan\phi_d / V \tag{7}$$

2.2 Reference Trajectory Generation

The goal of the trajectory generation algorithm can be formulated as follows: Given initial position components, $x(t_0) = x_0, y(t_0) = y_0$, an initial altitude, $h(t_0) = h_0$, final position components, x_f, y_f, and the final altitude, $h(t_f) = 0$, determine a γ^* and a ϕ_d^* so that

$$x(t_f) = x_f, y(t_f) = y_f \tag{8}$$

The trajectory design problem will be solved by transforming the NED velocity components into the wind frame; this procedure is carried out by defining

$$\dot{X} = \dot{x} - w_x(t), \dot{Y} = \dot{y} - w_y(t) \tag{9}$$

The resulting position components are computed by integrating Eq. (9):

$$X(t) = x(t) - \int_{t_0}^{t} w_x(\tau)d\tau, Y(t) = y(t) - \int_{t_0}^{t} w_y(\tau)d\tau \tag{10}$$

This transforms Eqs. (1) and (2) into

$$\dot{X}(t) = V\cos\gamma\cos\psi(t) \tag{11}$$

$$\dot{Y}(t) = V\cos\gamma\sin\psi(t) \tag{12}$$

In terms of the new variables, the initial conditions and final conditions, respectively, are

$$X_0 = x_0, Y_0 = y_0 \tag{13}$$

and

$$X_f = x_f - \int_{t_0}^{t_f} w_x(\tau)d\tau,\ Y_f = y_f - \int_{t_0}^{t_f} w_y(\tau)d\tau \tag{14}$$

where t_f being the flight time in the presence of wind, calculated by integrating Eq. (3),

$$t_f = -\frac{h_0}{V\sin\gamma^*} \tag{15}$$

It is now required to find two equations for γ^* and ϕ_d^*. To that end, for these calculations only, we neglect the transient response of the bank angle, since the servo time constant is much smaller than the flight time; thus, $\phi \approx \phi_d^*$ and (cf. Eq. (7))

$$\psi(t) = \psi_0 + \frac{g\tan\phi_d^*}{V}t \tag{16}$$

To complete the procedure, we integrate Eqs. (11) and (12) with $\psi(t)$ as in Eq. (16), and substitute $t = t_f$. This yields the desired equations for γ^* and ϕ_d^*:

$$X_f(\gamma^*) = X_0 + \frac{V^2}{g\tan\phi_d^*}\cos\gamma^*\left[\sin\left(\psi_0 + \frac{gt\tan\phi_d^*}{V}\right) - \sin\psi_0\right] \tag{17}$$

$$Y_f(\gamma^*) = Y_0 + \frac{V^2}{g\tan\phi_d^*}\cos\gamma^*\left[\cos\left(\psi_0 + \frac{gt\tan\phi_d^*}{V}\right) + \cos\psi_0\right] \tag{18}$$

where X_f and Y_f depend on γ^* through Eq. (14). Eqs. (17) and (18) render two algebraic equations for γ^* and ϕ_d^*. If a solution exists, then, at t_f, Eq. (8) is satisfied, and the parafoil will reach the target location with a miss distance induced by the servo time constant. This miss distance is defined by

$$\Delta r_f = \sqrt{\left[x(t_f) - x_f\right]^2 + \left[y(t_f) - y_f\right]^2} \tag{19}$$

The required bank angle is achieved by a proper servo deflection, while the flight-path angle is determined by changing the parafoil's drag-to-lift ratio (cf. Eq. (6)). The sign of the initial heading angle is determined according to the quadrant of the final position relative to the initial position:

$$\mathrm{sgn}(\psi_0) = \mathrm{sgn}\left[(x_f - x_0)(y_f - y_0)\right] \tag{20}$$

Sample results of the above process are given in Table 1. In this table, the flight time, t_f, γ^* and ϕ_d^* are calculated for a group of 10 parafoils, each designated with a different landing location, in the constant wind field $w_x = 5$ m/sec, $w_y = 10$ m/sec and the initial conditions $X_0 = -1000\,\mathrm{m}, Y_0 = -2000\,\mathrm{m}, H_0 = 3000\,\mathrm{m}$. The servo time constant is $\tau = 0.01\,\mathrm{s}$. It is seen that the trajectory design algorithm provides miss distances ranging from 0.9 m to 5.1 m, depending on the final coordinates.

Case #	x_f [m]	y_f [m]	γ^* [rad]	ϕ_d^* [rad]	t_f [sec]	Δr_f [m]
1	-1000	-500	-2.3055	0.0048	336.9266	3.5
2	-1000	0	-2.0956	0.0032	288.8697	2.4
3	-1000	500	-2.0087	0.0000106	276.0455	2
4	0	-500	-2.1244	0.0090	293.8917	2.1
5	0	0	-1.8153	0.0089	257.6664	0.9
6	0	500	-1.6558	-0.0014	250.9051	0.4
7	1000	-500	-2.3905	0.0098	366.3415	2.9
8	1000	0	-1.2187	-0.0088	266.3352	1.4
9	1000	500	-1.3332	-0.0026	257.2293	1
10	4000	4000	-0.6779	0.0026	398.6011	5.1

Table 1: Initial flight-path angle and desired bank angle for a group of 10 parafoils with different landing locations in the presence of a constant wind field

The reference trajectories of the parafoils are shown in Figure 1. The initial position is designated by an "o" and the final position is designated by an "x".

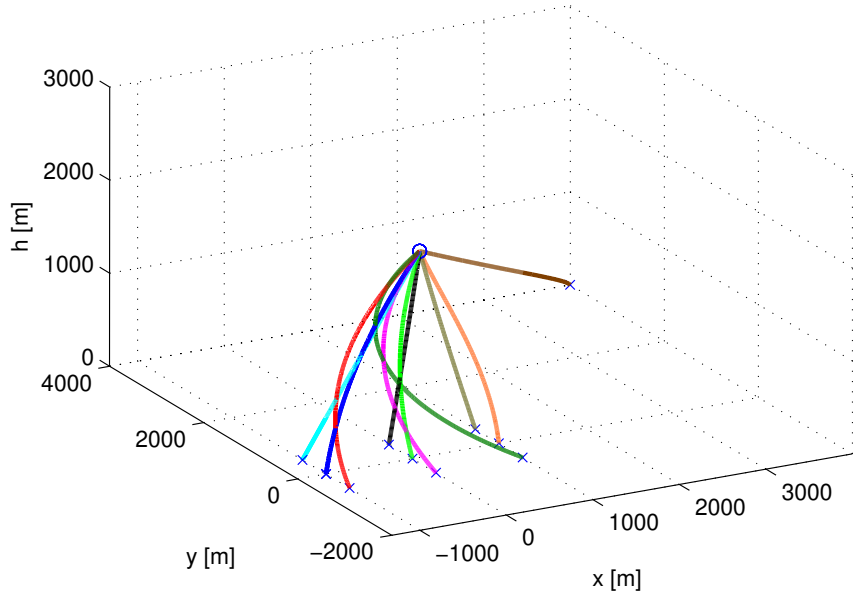

Figure 2: Sample parafoil reference trajectories in north-east-down coordinates

2.3 Guidance Algorithm

The purpose of the guidance algorithm is to steer the parafoils to the pre-determined reference trajectories (as determined by the initial flight-path angle and the steady-state bank angle) in the presence of initial release errors and perturbations, which cause the actual trajectory to deviate from the reference trajectory. Let $X(t), Y(t), h(t)$ denote the coordinates of the actual trajectory, and $X^*(t), Y^*(t), h^*(t)$ be a given reference trajectory. The nominal time-varying flight-heading is denoted by $\psi^*(t)$, and the actual heading angle is $\psi(t)$. Define the state variables deviations

$$\delta X(t) \triangleq X(t) - X^*(t),\ \delta Y(t) \triangleq Y(t) - Y^*(t),\ \delta h(t) \triangleq h(t) - h^*(t), \delta\psi(t) \triangleq \psi(t) - \psi^*(t) \quad (21)$$

If $\gamma^*(t)$, $\phi^*(t) \approx \phi_d^*(t)$ are the nominal flight-path and bank angles, respectively, and $\gamma(t)$, $\phi(t) \approx \phi_d(t)$ are the same angles defined for the actual trajectory, the differences

$$\delta\gamma(t) \triangleq \gamma(t) - \gamma^*,\ \delta\phi(t) \triangleq \phi(t) - \phi^* \quad (22)$$

become control inputs. The trajectory control is performed in closed loop, using the state-variables differences feedback. The parafoil is steered back to the nominal lateral and longitudinal coordinates, and the flight-heading angle difference is nullified. The altitude difference is regulated by an additional bias, so that the final altitude converges to zero at the given final lateral and longitudinal position.

To derive a closed-loop controller, small deviations from the reference trajectory are assumed. Linearizing Eqs. (3), (11), (12) about the nominal trajectory yields the linear time-varying model

$$\dot{\mathbf{x}}(t) = A(t)\mathbf{x}(t) + B(t)\mathbf{u}(t) \quad (23)$$

where $\mathbf{x}(t) = [\delta X(t)\ \delta Y(t)\ \delta\psi(t)]^T$ is the state vector and $\mathbf{u}(t) = [\delta\gamma(t)\ \delta\phi(t)]^T$ is the control vector. The system matrix, $A(t)$, and the input matrix, $B(t)$, are given by

$$A(t) = \begin{bmatrix} 0 & 0 & -V\cos\gamma^*\sin\psi^*(t) \\ 0 & 0 & V\cos\gamma^*\cos\psi^*(t) \\ 0 & 0 & 0 \end{bmatrix},\ B(t) = \begin{bmatrix} -V\sin\gamma^*\cos\psi^*(t) & 0 \\ -V\sin\gamma^*\sin\psi^*(t) & 0 \\ 0 & \frac{g}{V}(1+\tan^2\phi^*) \end{bmatrix} \quad (24)$$

A straightforward approach for designing a closed-loop trajectory controller for system (23) is neighboring optimal control [14].

Given a cost functional of the form

$$J = \int_0^{t_f} \mathbf{x}(t)^T Q\mathbf{x}(t) + \mathbf{u}^T(t)R\mathbf{u}(t)dt + \mathbf{x}^T(t_f)M_f\mathbf{x}(t_f) \quad (25)$$

with Q being a 3×3 state penalty matrix, R a 2×2 control weight matrix and P_f a 3×3 terminal weight, a stabilizing feedback minimizing (25) is given by

$$\mathbf{u}(t) = -R^{-1}B^T(t)M(t) \quad (26)$$

where $M(t)$ is a solution of the differential matrix Riccati equation

$$-\dot{M}(t) = A^T M(t) + M(t)A + M(t)B(t)R^{-1}B^T(t)M(t) + Q \quad (27)$$

Finally, the differential flight-path angle resulting from Eq. (24) is augmented by a constant bias, b, guaranteeing that $h(t_f) = h^*(t_f) + \delta h(t_f) = 0$:

$$\delta\dot{h}(t) = V\cos\gamma^*\left[\delta\gamma(t) + b\right] \tag{28}$$

To illustrate the performance of the said guidance law, we performed a simulation of the nominal trajectory described by Case 3 in Table 1 with $\delta X_0 = 5$ m, $\delta Y_0 = 5$ m, $\delta h_0 = 4$ m, $\delta\psi_0 = 0.001$ rad . The simulation results are shown in Figure 3, depicting the three differential position components and the flight-heading angle. The magnified state variables show the relatively short transient of the closed-loop control. The feedback regulates the lateral and longitudinal position components and the flight-heading angle. The altitude regulator is in fact a terminal controller, nullifying the altitude difference at impact, so that the miss distance is identical to the value given in Table 1.

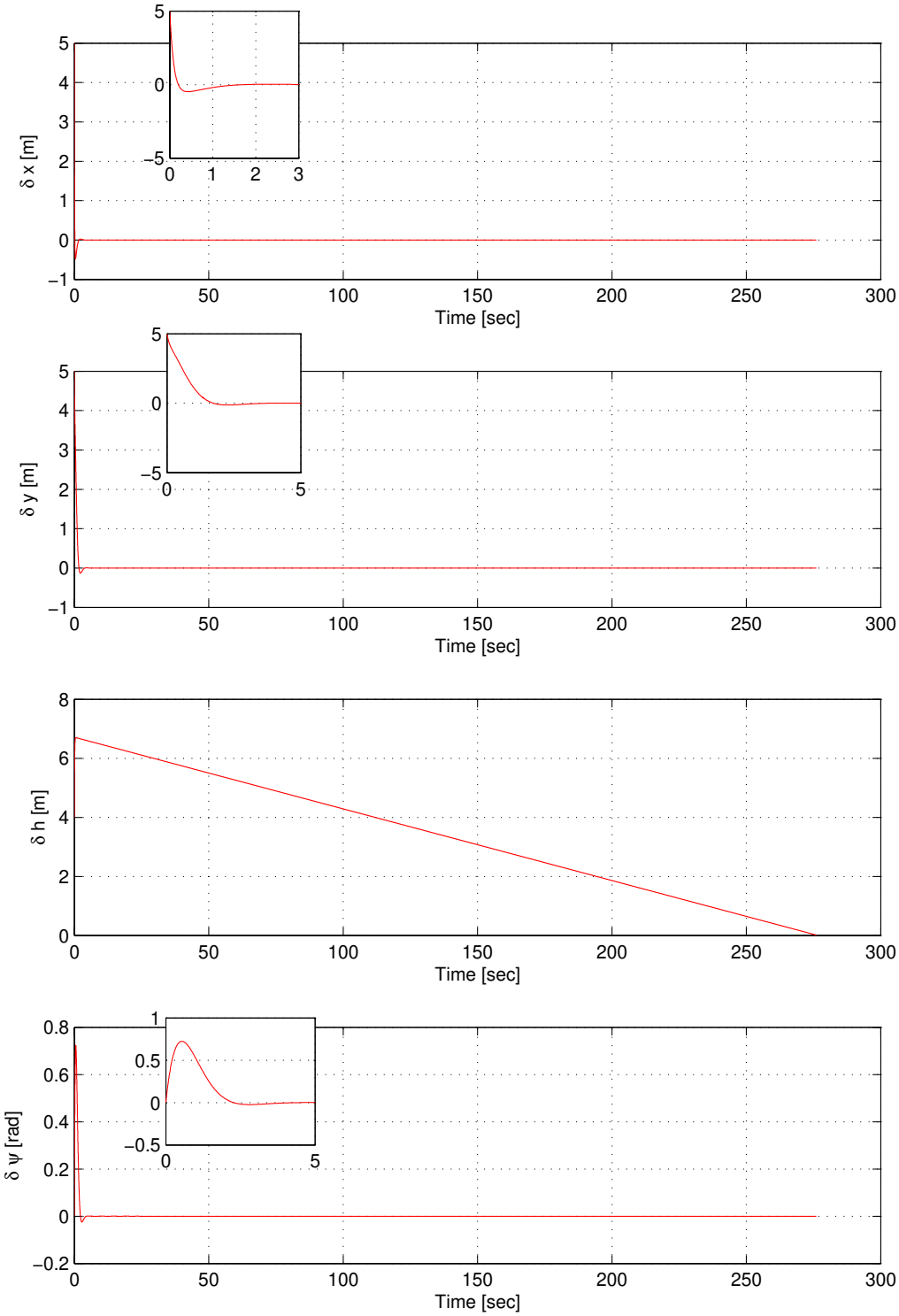

Figure 4: Guidance using neighboring optimal trajectory control. After a transient of about 2 s, the X and Y components converge to zero, while the altitude difference is nullified at impact

3. Parafoils Mission Planning

In this section, we develop mission planning and task management algorithms for the guided parafoils described in the previous section. This includes a mechanism for target reassignment. To illustrate the importance of task management, we begin with a simple motivating example.

3.1 Simple Motivating Example

Assume that the probability of a given parafoil to succeed in an airdrop mission (i.e., deliver the payload to the target with some pre-defined allowed miss distance) is *p*, and that the probability to fail is $q = 1 - p$. The target of parafoil P_h has the highest priority and the remaining *n*-1 parafoils have targets with lower priorities. Let q' be the probability that P_h failed in an altitude that still allows a replacement by another parafoil. We are interested in evaluating the probability that the target of P_h will get the needed supply. To that end, we distinguish between two cases:

1. There is no communication link between the parafoils. In this case, the probability that P_h reaches the target is *p***.**
2. There exists a communication link between the parafoils. In this case, the probability that P_h or a replacing parafoil reach the target is $p + q'(1 - q^{n-1})$.

Thus, if there are, e.g., 9 parafoils in the group and $p = 0.9,\ q = 0.1,\ q' = 0.1$, then the probability that the parafoil P_h will hit the target in the absence of communication is 0.9, while the probability that some parafoil will reach the initial target of P_h is 0.981.

This simple example highlights the importance of task management, to be dwelt upon in the remainder of this section.

3.2 The Interactive Design Tool

The process of planning aerial support to regions hurt by natural disasters requires detailed planning. During the planning phase, the operation manager (OM) is required to identify the geographic locations where the survivors concentrate, and the type of support needed by these survivors. The next stage in the mission planning process is the selection of the parafoil release coordinates based on the topography, flight corridors, the total weight of the payload, wind speed and direction, and parafoils state.

To facilitate the mission planning, we have developed an interactive design tool (IDT). Figure 5 presents a screenshot of the IDT. The OM first identifies the coordinates of targets that require support and the type of support. This is done by selecting a colored thumbtack and by fixing it onto the interactive map. It is possible to fix several thumbtacks in each spot according to the actual needs.

The next step is determining the parafoil release coordinates. *Release coordinate* are defined as the center of an *airdrop envelop* covering a given set of thumbtacks, as shown in Figure 5. The lines that connect the center of the airdrop envelope to the thumbtacks indicate that there exists a reference trajectory from the release position to the target. The radius of the airdrop envelope is computed based on Eqs. (17)-(18) given $h(t_0)$.

In case of airdrop envelope overlap (thumbtacks can be included in more than a single envelope), the IDT will automatically select the airdrop position that creates the shortest path for every thumbtack. As the mission design process is iterative, the airdrop position will be updated automatically according to the OM's online updates.

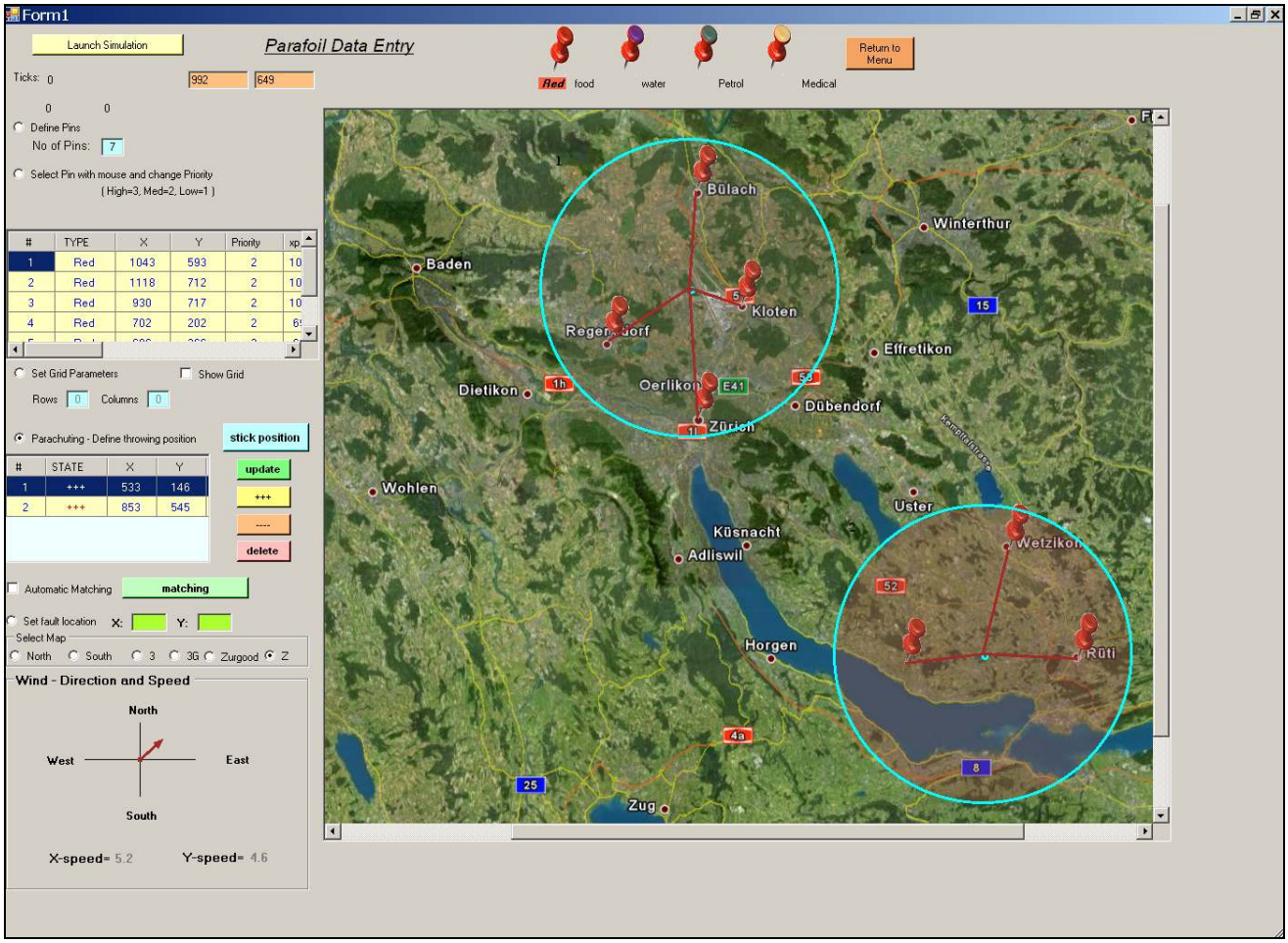

Figure 5: Snapshot of the mission planning screen. This interface provides an interactive design tool for prioritizing target locations in need of various support types and calculate the airdrop envelopes

3.3 Task Management Algorithm Logic

Every parafoil, P_i, carries a single payload, to be delivered to a predefined target (or, equivalently, a task) T_i. A target is characterized by the following attributes:

1. The (x_f, y_f) coordinates;

2. Required payload: food, water, medical supply and petrol (possibly, more than one payload can be delivered to a given target);
3. A priority, denoted by $\Pi(T_i) = \Pi_i$, where $\Pi_i = \{1, 2, 3\}$ and 3 denotes the highest priority.

The main goal of the task management algorithm is to minimize the loss of payload in case of parafoil failure or fault. A failure of a mechanical element, a failure of the navigation unit or winds may prevent a parafoil from landing at its predefined target position. However, even though a parafoil may be unable to accomplish its original mission, it can divert its gliding direction to a closer target or to a target located at a different heading. This process is managed by the task management algorithm, depicted by the state diagram shown in Figure 6.

The task management algorithm, implemented onboard each parafoil, periodically executes an *audit* process. This process evaluates the parafoil ability to complete its original mission. If the audit process determines that the parafoil is unable to complete its mission, the parafoil will commence on a recovery procedure. This procedure starts with a *Data Collection* step (Figure 6), in which the parafoil collects data about other parafoils with identical payloads. As soon as this step terminates successfully, the parafoil starts the *Swap Planning* step (Figure 6). *Swap* is defined as a bijective permutation of the targets within a group of parafoils, a permutation that keeps an objective function of the form

$$J = \sum_i \Pi_i \tag{29}$$

unchanged.

In some cases, however, it is impossible to swap targets, so low priority targets must be abandoned in favor of higher priority ones. The *Replace* procedure is used for finding a solution for target allocation in such cases instead of the Swap mechanism. We define

Replace as follows: let $J_G = \{J_1, J_2, \ldots, J_n\}$ be the group of possible values of the objective function *J* calculated by abandoning a target. Then *Replace* performs target re-allocation so that

$$J^* = \max J_G \tag{30}$$

However, Swap is the preferable procedure, as it overcomes the difficulties of faulty parafoils without affecting the targets needs.

The *Glide* state, shown in Figure 6, constantly monitors the trajectory by comparing pre-calculated reference points on the reference trajectory to the actual trajectory. The Glide process initiates corrective maneuvers according to Eq. (26). If the deviation of the actual trajectory relative to the reference trajectory crosses a given threshold (this threshold is determined by servo saturation), Glide will declare the parafoil as *faulty*. Another event that will lead to a declaration of a parafoil as faulty is when Glide gets a malfunction indication from one of the hardware elements. The type and severity of the fault are major factors in deciding weather a parafoil will be able to participate in either Swap or Replace, which in turn depend on the successful termination of the Data Collection process.

The ability to perform Swap or Replace is evaluated using a unified mechanism – the potential Swap graph (PSG). Using the ad-hoc communication infrastructure, to be described shortly, a faulty parafoil collects data from the surrounding parafoils and calculates J^* using the breadth-first search algorithm described in the Appendix.

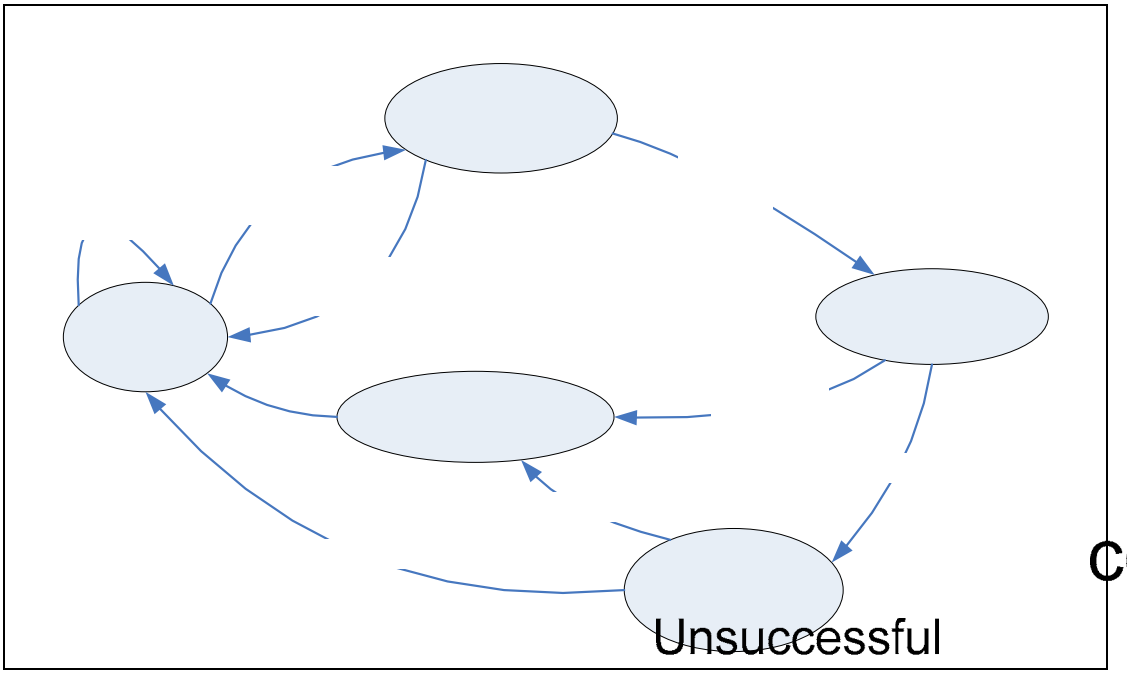

Figure 6: Task management state diagram

3.4 Swap and Replace Decision Logic

Table 1 presents the basic rules used for deciding whether a given parafoil P_r (whose target is T_r) swaps or replaces a faulty parafoil P_f (whose target is T_f).

	Swap	**Replace**
Payload	The payload of P_r is of the same as the payload of P_f	
Priority	N/A	$\Pi(T_r) < \Pi(T_f)$
Landing	P_r is capable of changing its heading and land in the original (x_f, y_f) of P_f and vice versa.	P_r is capable of changing its heading and land in the original (x_f, y_f) of P_f.
Alternative selection	In case that a group of parafoils is able to replace/swap P_f, the task management process will select the parafoil with the shortest trajectory to T_f.	

Table 1: Basic Swap and Replace rules

3.5 Swap and Replace Illustrative Examples

In this section, we will illustrate Swap and Replace using a few simple examples. To begin, Figure 7 depicts a 2-way and a 3-way parafoil Swap scenarios. In Figure 7-A, the audit process determines that P_1 is unable to complete its mission successfully. A successful Swap procedure must then exchange between T_1 and T_2. This procedure does not affect the targets, as the two parafoils carry the same type of payload.

Figure 7-B shows a 3-way parafoil swap procedure. As in the previous case, the audit determines that P_1 is unable to complete its mission, and that P_2 is unable to swap tasks with P_1 as before; however, it is possible to create a cycle of parafoil diversions that will successfully satisfy the requirements of each target.

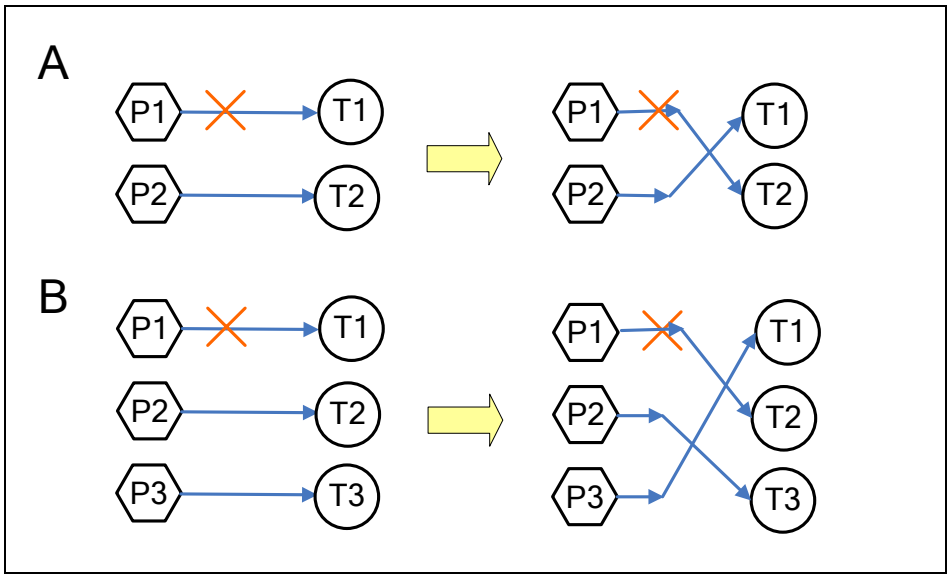

Figure 7: Schematic description of a 2-way and a 3-way parafoil Swap processes. In each case, the targets get their required supplies, but not by the original parafoils assigned to those targets.

To illustrate the Replace procedure,

Figure 8 presents two scenarios. A simple case is presented in

Figure 8-A. P_1 is unable to complete its mission and Swap in not applicable. In this case, as $\Pi(T_1) = 2$ and $\Pi(T_2) = 2$, T_2 is abandoned in order to supply the more urgent needs of T_1.

Figure 8-B presents a more complicated case wherein a 3-way parafoil Replace is requiered.

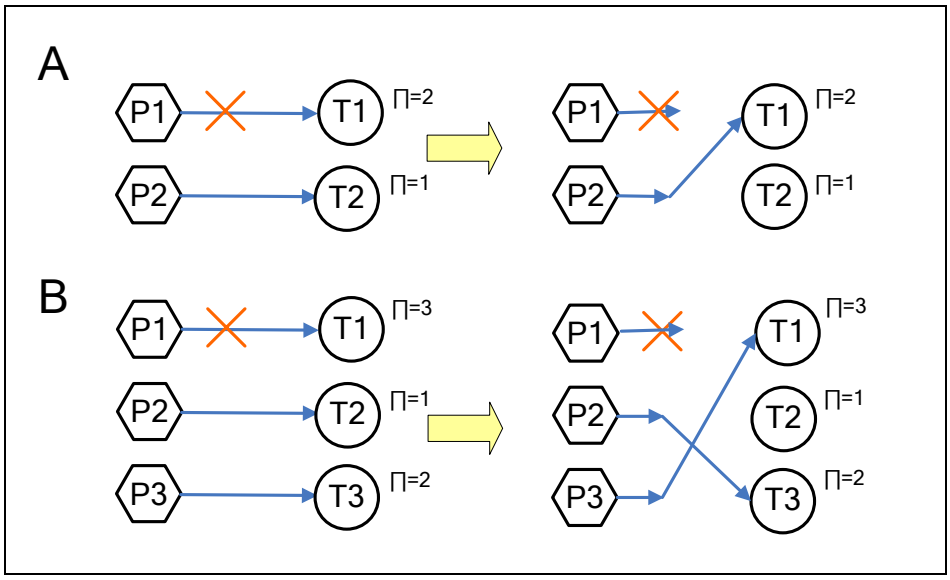

Figure 8: Schematic description of a 2-way and a 3-way parafoil Replace. This process is responsible for reassignment of targets in case of parafoil failure that cannot be remedied using Swap.

In the example presented in Figure 9, an arc $P_x \rightarrow P_y$ indicates that P_y can replace P_x. After analyzing this potential Swap graph (PSG), P_f determines its ability to be replaced by another parafoil or, preferably, to swap targets with another parafoil.

Figure 9-A present the decision-making process after P_f declares itself a faulty parafoil and completes building the PSG. The following replacements are applicable:

1. P_1 will replace P_f as $\Pi(T_1) < \Pi(T_f)$. In this case, T_1 will be abandoned.
2. A chained replacement procedure where P_3 replaces P_f and P_4 replaces P_3, as $\Pi(T_4) < \Pi(T_3) < \Pi(T_f)$. In this case, T_4 will be abandoned.

The algorithm will prefer the first alternative as the number of diversions in the first case is 2 and in the second case is 3 and $\Pi(T_1) = \Pi(T_4) = 1$.

Figure 9-B presents a different situation. P_f declares itself a faulty parafoil, as it cannot glide and land at its original target. If it can change its target and glide successfully to T_2, a cyclic swap process will be initiated so P_1 will replace P_f, P_5 will replace P_1, P_2 will replace P_5 and P_f will replace P_2. Note that in this case the priority is irrelevant.

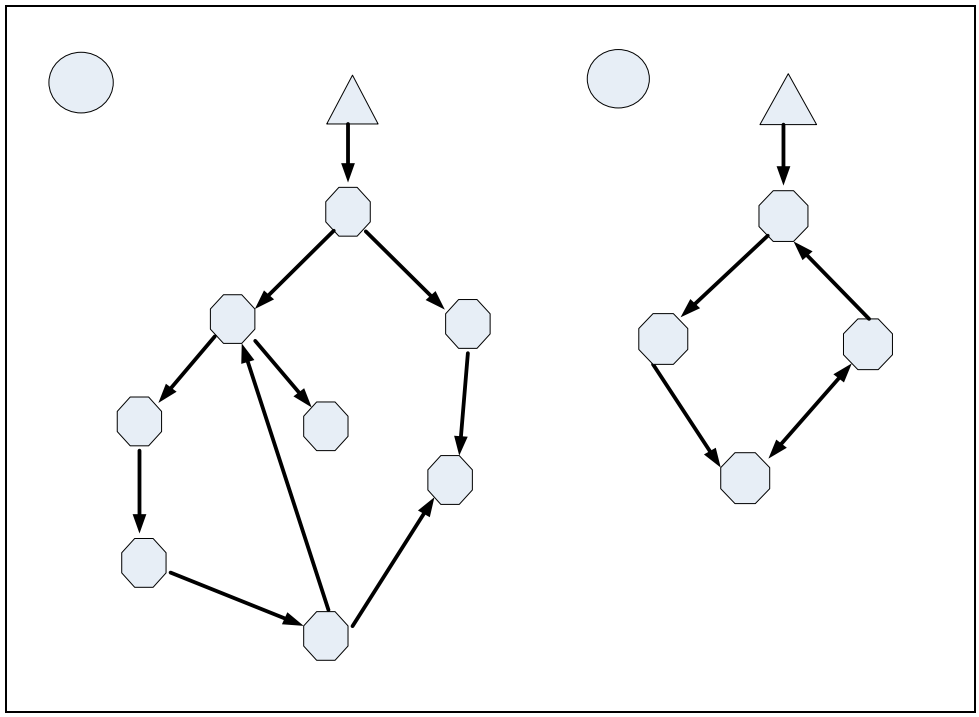

Figure 9: Potential Swap graph examples

A

Tf

4. Communication

To purpose of the inter-parafoil communication system is to enable the Swap and Replace functions discussed above, so that the correct type of support will eventually reach the pre-designated targets as accurately as possible.

4.1 The Communication Model

The communication system is responsible for transferring applicative data among parafoils. The communication model is composed of 3 tiers: the Medium Access Control (MAC) layer, which is based on a 802.11[15] standard; the Metrical Routing Algorithm (MRA) [1] layer, which runs on top the MAC layer and is responsible for routing messages between parafoils; and the applicative layer (AL), which is the upper-level of the communication system and is responsible for the applicative content of the messages transferred between every two or more nodes (parafoils) which are network members.

4.1.1 Radio Propagation Model (RF)

The RF model is used to quantify radio propagation between any two nodes (parafoils) in the theater. We assume that the transmitters use isotropic antennae that radiate uniformly in all directions. This type of antenna is often used as a reference for antenna gain in wireless systems. It uses the Effective Radiation Power (ERP) formulas, given below.

The loss factor between a transmitting and a receiving node, denoted by *loss* and measured in units of dB, assuming RF propagation using a free-space transmission between two points at a distance *d*, is given by:

$$loss = 92.5 + 20 \times \log(d \times f) \tag{31}$$

where *d* is in units of km and the transmission frequency, *f*, is measured in GHz. Similarly, the loss factor for RF propagation of antennae that are near the ground is given by:

$$loss = 40 \times \log(d) - 20 \times \log(Ht \times Hr) \tag{32}$$

where d is distance between antenna in meters and H_t and H_r is the altitudes of the transmitter and the receiver in meters, respectively, above the ground.

Similarly to the management of the RF model, it is possible to manage the links bandwidth and packet transmission rate. The bandwidth required to maintain the communication links among parafoils is minimal, and does not exceed 1000 bits/second.

4.2 Metrical Routing Algorithm (MRA)

The MRA algorithm [1] is used for connecting the parafoils into communication networks. It is also used for transferring queries among parafoils, change the tasks of parafoils and control the information flow.

The MRA attempts to connect the parafoil group $G = \{P_1, P_2, ..., P_N\}$ by a minimal set of rooted trees that preserve geographical distances; viz. distances on the rooted trees are usually proportional to the distances of $P_1, P_2, ..., P_N$ in a given theater.

More formally, let $G(t)$ be the graph at time t wherein each two nodes P_i, P_j have an edge in $G(t)$. The MRA algorithm attempts to cover $G(t)$ by a minimal set of spanning trees. The rooted trees created by the MRA algorithm can be naturally used for both distributed computing (of, e.g., the applicative layer) as well as for communication and data propagation tasks.

4.2.1 The MRA Protocol

The MRA protocol presented herein is a hybrid ad-hoc protocol in the sense that some traffic control is used to maintain the mapping of the communicating nodes. The small overhead of the MRA protocol used to maintain the mapping is a worthy investment, as the MRA is capable of handling successfully a demanding traffic load under a high node density and fast node movement. The MRA organizes the nodes in rooted trees in order to find short session paths between nodes on the tree. The algorithm attempts to minimize the number of trees by fusing separate adjacent trees into a single tree. As long as any

node in one tree is not in transmission range of any node in the other trees, the trees will function autonomously. As soon as a radio connection is created between two nodes, the trees will be fused into a single tree.

All nodes run the same protocol implementing the MRA. As nodes may emerge, disappear and move in or out of range of other nodes, there is need to update the trees. A primary goal of the algorithm is to identify these changes and adapt the tree structure to the new state. In the following discussion, we shall present an elaborate description of the MRA protocol.

The MRA algorithm organizes the nodes in the field in rooted trees. Only nodes that belong to the same tree can create sessions among themselves. To ensure maximal connectivity, all nodes will try to organize themselves in a single tree. Every node in the field has a unique Parafoil-ID (PID) (similar to a phone number or an IP address) and virtual coordinates that may change depending on the changes in the tree structure. Every tree is identified by a *tree name*, which is the PID of the root node.

Nodes periodically send beacons, termed *hello* messages. Every node that receives a beacon checks whether the node that sent the beacon belongs to a different tree. If the nodes belong to different trees, they initiate a fusion process that fuses the separate trees into a single tree. The fusion protocol should satisfy the following properties:

1. The protocol should not cause active sessions to break;
2. Eventually (assuming no dynamic changes occur) all trees with nodes within transmission range must fuse into a single tree;
3. When two trees are being fused, most updates should be made to the nodes of the smaller tree (in terms of the number of nodes);
4. The protocol should maximize the number of nodes that migrate from one tree to another in every step (yielding parallel fusion);
5. The protocol is fully distributed with no “central" bottlenecks, namely it is defined at the level of pairs of nodes.

Initially, every node forms a separate tree of size 1. Every node in the tree can autonomously migrate to a neighboring tree regardless of the node position in the tree. The migrating node gets new coordinates in its new tree according to the node's new position. Naturally, when a node migrates from one tree to a new tree, it may carry along its neighboring nodes (since it belongs now to a bigger tree). In the macro view, the migration of the single nodes resembles a fusion of smaller trees into bigger ones.

Figure 10 illustrates two stages of the tree fusion process: The initial state and the final organization into trees (assuming no significant node movements occurred during this process).

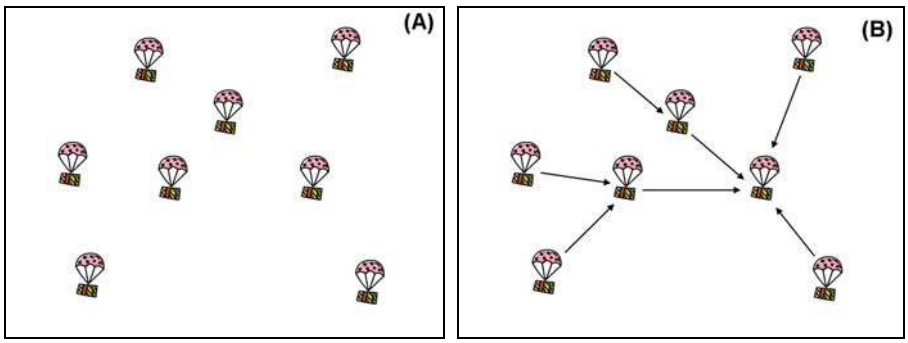

Figure 10: Tree formation process

Fusion of two trees is a parallel process, where at any given stage one or more nodes of the smaller tree join the larger tree, as depicted in Figure 11A and Figure 11B. The implementation of the mission planning algorithms is based on this tree structure. Every tree autonomously runs these algorithms as it does not have communication with other trees. Existence of such communication will initiate a merge process that will result in a single tree.

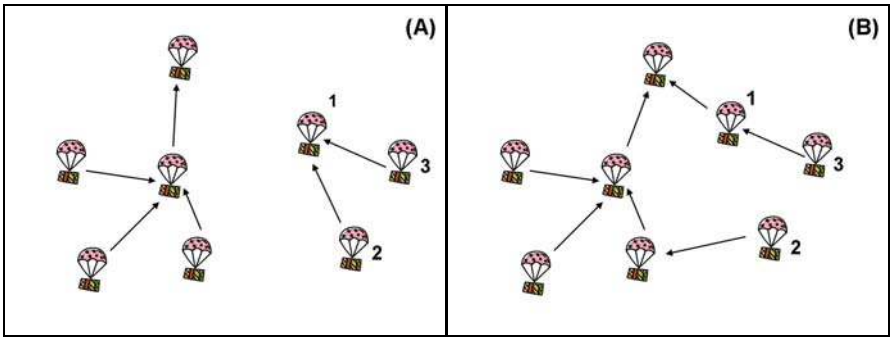

Figure 11: Fusion of trees

4.3 Implementing Replace and Swap

The same AL package runs in all parafoils, exchanging the following types of messages:

1. A direct message, which is sent from the source node to a target. Sending a direct message implies that the source node will be able to identify explicitly the PID of the target parafoil.
2. A multicast message, which is sent from a source node to a group of nodes that are identified by a common identifier.
3. A broadcast message, which is sent from a source node to all nodes in the network. The source node is unaware of the number of nodes that receive this message or the number of nodes that create the network.

Regardless of the message type, the nodes are unaware of the path that a message passes on its way from the source to the target nodes. The ad-hoc infrastructure is responsible for transferring a message transparently to the target node. Moreover, the parafoils do not have a leading or a managing node. A parafoil assessing that it is unable to complete its mission, autonomously initiates either Swap or Replace.

Figure 12 shows the messages flow that takes place when a faulty parafoil is looking for a replacing parafoil. The faulty parafoil (P_f) sends a broadcast message to all parafoils in the network. In this message P_f indicates its task, priority and own PID. Every parafoil that receives this message performs the following actions: (i) it continuous the broadcast

process by sending the message to its parent and offspring in the tree (its does not send the message to the sender to prevent loops) and (ii) it evaluates, according to the target and its priority, weather it can replace the faulty parafoil. If its target's priority is higher or equal to the originator's target priority, or its own task differs the faulty parafoil's task, it discards the message (P_2 in Figure 12). If its priority is lower than the originator's priority and its task is identical to the originator's task, then it returns a direct message to the initiating parafoil (P_3 in Figure 12).

In the direct message returned to the originator, the parafoil indicates its target priority and location and its own PID. The originating parafoil collects the answer messages arriving from parafoils in the network until the "wait" timeout expires. The expiration of the timer indicates to the originating parafoil that it should not wait any more for late or lost messages but rather start analyzing the answers immediately. The analysis of the answers is aimed at selecting the most suitable parafoil to replace the faulty parafoil.

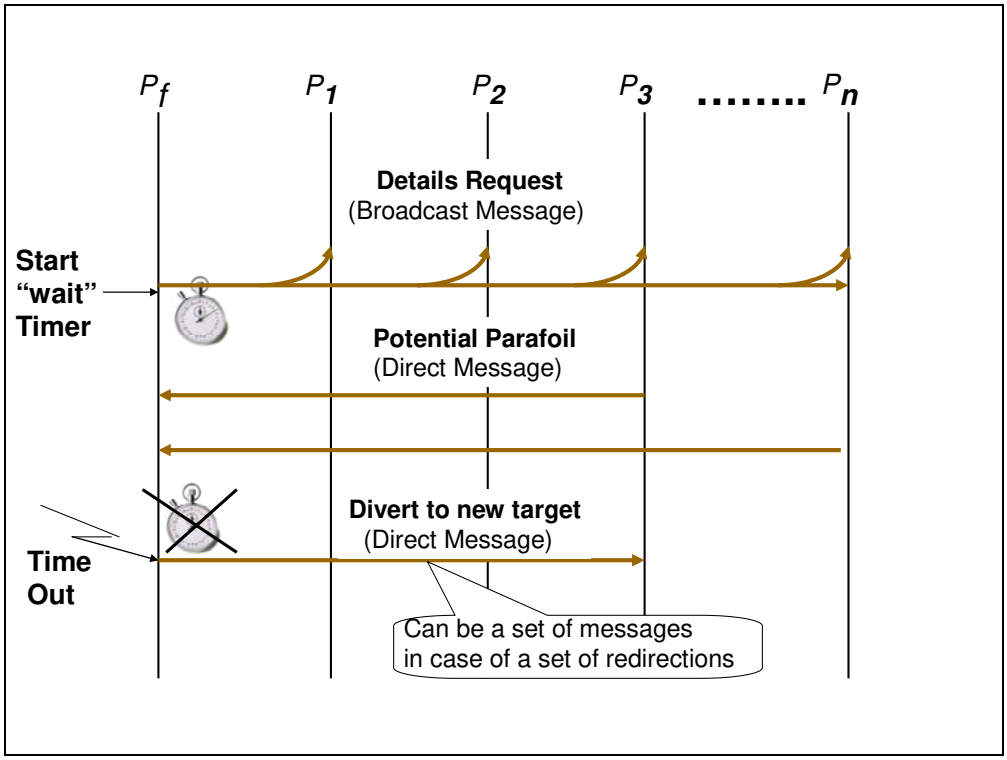

Figure 12: Alternate parafoil selection process

Note that a search for a replacing parafoil may end with a false result, which means that the falling parafoil did not find a replacing parafoil.

4.4 Communication Load Analysis

The inter-parafoil communication is divided into two categories: idle time communication and recovery time communication. The idle time load is created by the nodes in order to preserve the ad-hoc network. This ensures that when needed, the faulty parafoil will be able to start Swap or Replace immediately. The idle load on every node consists of inbound traffic (load on the receiver) and the outbound traffic (load on the transmitter). Table 2 presents the messages and the frequency of sending and receiving the idle load messages.

	Inbound Traffic	**Outbound Traffic**
Hello messages	A message is received every 0.04 sec (assuming maximum 10 neighbors)	Every node transmits a message every 0.4 sec
Registration messages	A message is received from up to 5 offspring every 5 sec	Every node transmits a message every 5 sec

Table 2: Idle time messages and frequency of messages

The recovery from a fault requires the transmission of extra messages in a very short time. This traffic (presented in Figure 12) consists of the following messages:

1. A single message broadcast , *details_request*, generated by P_f.
2. A set (< 10) of *potential_parafoil* direct messages
3. A set (< 10) of *divert_to_a_ new_ target* direct messages.

The recovery time presents the peak requirements from the communication network and is generated mostly by P_f and less on the parafoils that participate in the search for a replacing parafoil. Table 3 presents the maximal calculated load in bytes/second during idle time and recovery time.

	Idle time traffic	Recovery time - extra traffic	Total
Inbound	1800	800	2600
Outbound	170	800	970

Table 3: Maximal inbound and outbound traffic load (bytes/s)

The given numbers represent very moderate requirements that can be satisfied by any basic radio equipment.

5. Simulation Environment and an Illustrative Example

In this section we provide a short overview of the simulation environment and discuss an illustrative example for the technology developed in the previous sections.

5.1 Simulation Environment

The simulation environment is based on the Interactive Flexible Ad-Hoc Simulator (IFAS) [10]. Originally, this simulator was designed to simulate ad-hoc networks. It was expanded to support cooperative parafoils simulation.

The IFAS was planned to be used for 3D simulations and it thus implements a 3D radio propagation model, including physical obstacles (such as buildings) that interpose between transmitters. The simulator includes a set of functions that can be used during the execution phase to support the online analysis, and hosts a set of tools that can be used on the output log files created during the run. An important element in the simulator is the implementation of every parafoil in the theater as a dedicated process. Every process uses the communication stack to communicate with other processes using the MRA protocol.

The IFAS provides an interactive, highly visual display wherein the user can view each simulated node and change its settings. This display includes additional views of

parameters and control data. Figure 13, presents, for example, a part of a window used to define graphically the wind speed and direction. Figure 14 shows some snapshots from the main simulation screen.

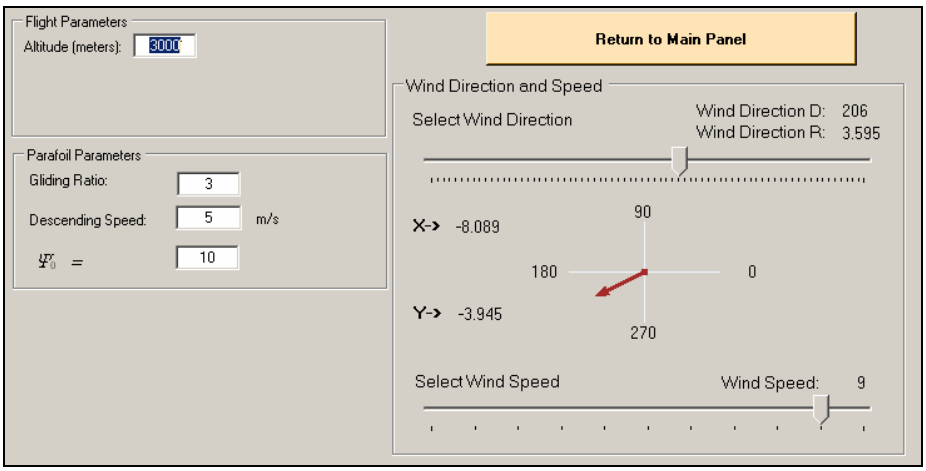

Figure 13: Wind direction and speed definition

5.2 The Scenario

In Figure 14 we present a scenario of the design, analysis, glide and recovery steps. In this scenario, we drop 5 parafoils carrying identical payloads required in 5 different locations. Figure 14-A presents the fist phase of the airdrop process.

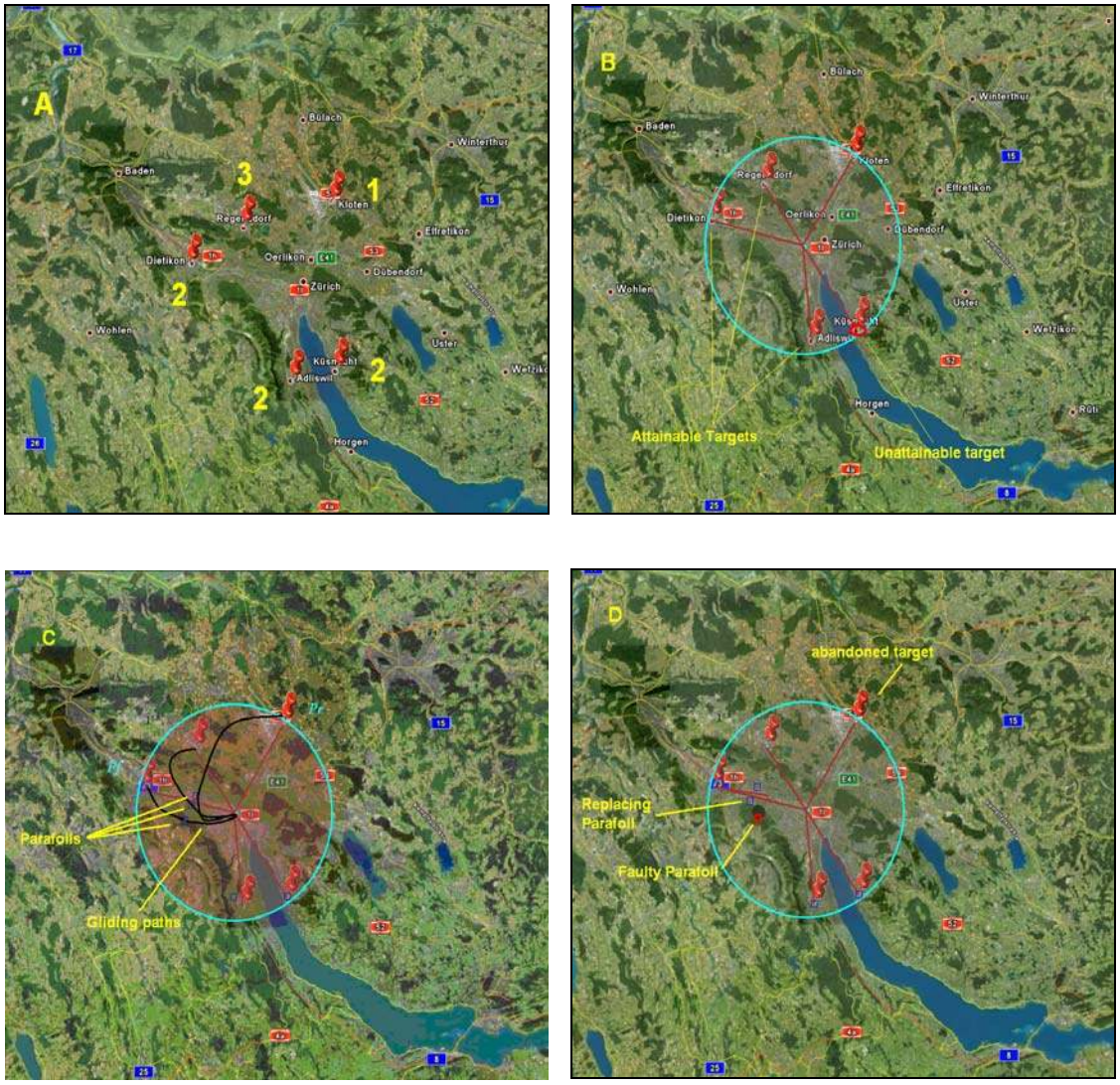

Figure 14: Snapshot of airdrop and parafoil recovery scenarios

The OM defines the airdrop position by fixing thumbnails on the target positions. The number near every thumbtack represents the target priority (the default value is 2). In our example, 3 targets have priority 2 while the other 2 targets have priorities 1 and 3.

Figure 14-B presents the next step of the dropping design process. The operator marks the preferred airdrop position, and the simulator analyses the ability of every parafoil to land at its target location.

Figure 14-C shows the gliding process of the parafoils. The parafoils are presented as small squares that move along their paths toward the target. Note that the black line does not really appear on the map and was added to the screenshot. An uninterrupted gliding process will end when the parafoils reach their targets.

Figure 14-D presents a case where one of the parafoils fails. We define manually on the simulator screen a fault position. This position is marked by a gray circle. A parafoil that glides from the airdrop position and enters the gray circle declares itself as a faulty parafoil and initiates a recovery process. As soon as the simulation is activated, the parafoils, which are represented by small squares, start moving from the airdrop position and glide along their trajectory to the target. In our case, the parafoil targeted to the leftmost target fails. A swap process in not applicable and the only solution for this case is to abandon one target with the minimal priority.

Figure 15 depicts the 3D trajectories of the failing parafoil, P_f, and the replacing parafoil, P_r, initially targeted to T_f and T_r, respectively (P_f and P_r are presented also in Figure 14-C). This figure shows the altitude (h) and the time (t) in seconds of every event. The parafoils start gliding simultaneously from an altitude of $3000\,\mathrm{m}$. P_f fails after 472 s at an altitude of $h = 1612\ \mathrm{m}$. It initiates the Replace process. As the priority of T_f is higher than the priority of T_r and P_r can replace P_f, P_r diverts its original trajectory at an altitude of $1808\ \mathrm{m}$ to an alternative trajectory that terminates at T_f after 1170 s.

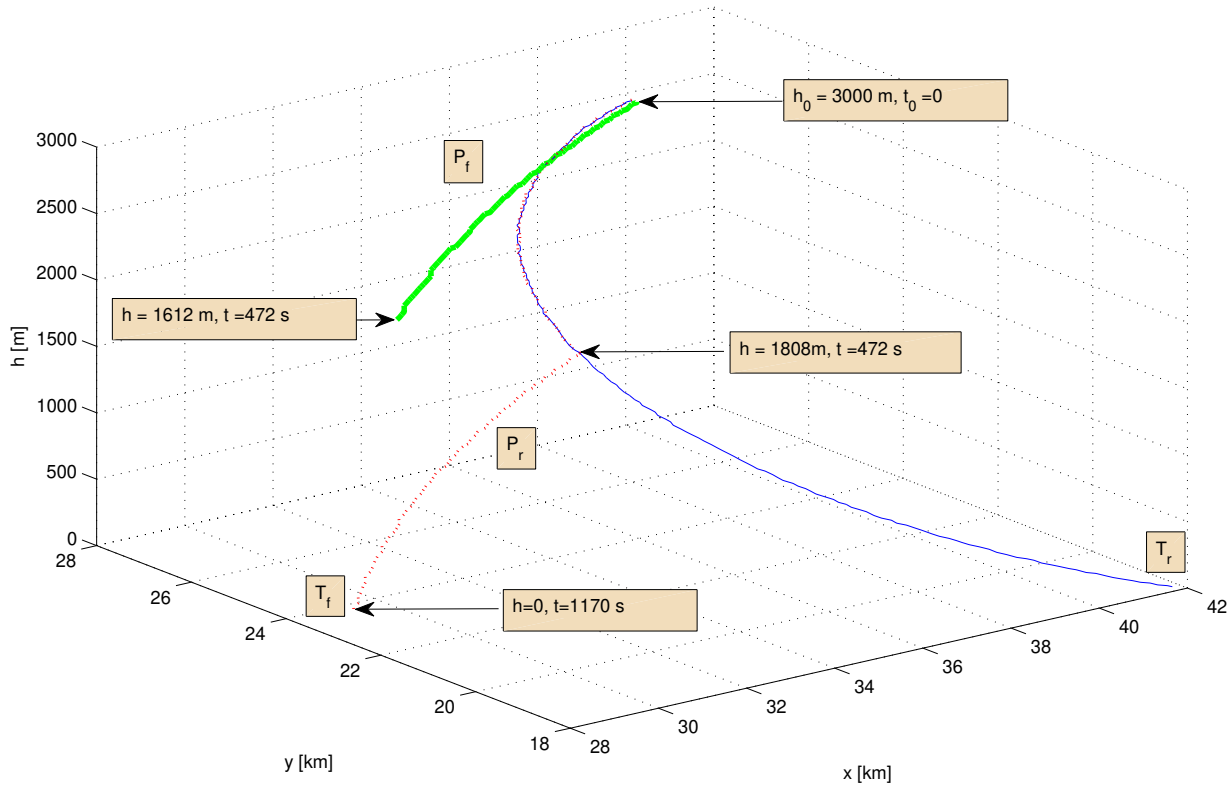

Figure 15: Gliding trajectories of the replacing (P_r) and failing (P_f) parafoils. The replacing parafoil engages the original target of the failing parafoil.

6. Conclusions

In this paper, we presented an innovative approach for handling a humanitarian aid mission based on a group of autonomous cooperative parafoils. The approach combined a guidance algorithm, an interactive planning tool used to plan the mission and a task management logic that upgraded the parafoils from individual entities to a cooperative system that can autonomously handle unexpected events and failures.

The main conclusion is that by combining a communication layer into the parafoils, the conservative approach of individual parafoils focused on their predefined targets can be leveraged to yield a reactive group that is capable of handling unexpected events.

Moreover, within the limits of a high-altitude airdrop mission that uses gliding parafoils without propulsion, the technology presented here results in significant improvements and will be useful in humanitarian airdrop missions.

Appendix: The Breadth-First Search Algorithm

The breadth-first search (BFS) algorithm is an uninformed search method that aims to expand and examine all nodes of a graph systematically in search of a solution. It exhaustively searches the entire graph without a-priori considering the goal. The pseudocode of the BFS algorithm, as implemented in our system, is given below.

Input: Graph $G= (V, E)$
Output: Graph G with its vertices marked by consecutive integers in the order they have been visited by the BFS traversal. A vertex v with the value 0 indicates that v is unvisited.

```
count ←0
For each v∈V do
     If value(v) = 0
          bfs (v)
     endif
endfor
```

bfs(v) procedure

Visits all unvisited vertices connected to v by a path, and assigns a number in the order visited using the global variable `count`.

```
count  ← count +1
add v to queue
while the queue is not empty do
     for each vertex w in V adjacent to the front vertex do
          if w is marked with 0
          count ← count+1
          add w to the queue
         endif
     remove the front vertex from the queue
     endfor
endwhile
```

Acknowledgments

This research was supported by the European Sixth Framework Program through the FastWing CL project and by the Gordon Center for Systems Engineering of the Technion.

References

[1] Dudek, G., Jenkin, M., Milios, E., and Wilkes, D., "A Taxonomy for Multiagent Robotics," *Autonomous Robots*, Vol. 3, 1996, pp. 375–397.

[2] Reif, J., and Wang, H., "Social Potential Fields: A Distributed Behavioral Control for Autonomous Robots," *Robotics and Autonomous Systems,* Vol. 27, 1999, pp. 171-194.

[3] Mataric, M. J., "Behavior Based Control: Examples from Navigation, Learning, and Group Behavior," *Journal of Experimental and Theoretical Artificial Intelligence* , Vol. 9, No. 2-3, April 1997, pp. 323-336.

[4] Reynolds, C. W., "Flocks, Herds, and Schools: A Distributed Behavioral Model," *Computer Graphics*, Vol. 21, 1987, pp. 25–34.

[5] Passino, K., Polycarpou, M., Jacques, D., Pachter, M., Liu, Y., Yang, Y., Flint, M., and Baum, M., "Cooperative Control for Autonomous Air Vehicles," *Proceedings of the Cooperative Control Workshop, Florida*, December 2000.

[6] Parunak, H. V. D., Purcell, M., and O'Connell, R., "Digital Pheromones for Autonomous Coordination of Swarming UAVs," *Proceedings of the AIAA First Technical Conference and Workshop on Unmanned Aerospace Vehicles*, Vol. 3446, 2002.

[7] Cunningham, C. T. and Roberts, R. S., "An Adaptive Path Planning Algorithm for Cooperating Unmanned Air Vehicles," *Proceedings of the IEEE International Conference on Robotics and Automation*, Seoul, South Korea, May 2001.

[8] Rasmussen, S., Chandler, P., Mitchell, J. W., Schumacher, C., and Sparks, A., "Optimal vs. Heuristic Assignment of Cooperative Autonomous Unmanned Air Vehicles," *Proceedings of the AIAA Guidance, Navigation & Control Conference*, Austin, TX, 2003.

[9] Ben-Asher, Y., Feldman, M., Feldman, S., "Ad-Hoc Routing Using Virtual Coordinates Based on Rooted Trees", *Proceedings of the SUTC 2006, the IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*, Taichung, Taiwan July 2006.

[10] Ben-Asher, Y., Feldman, M., Feldman, S., and Gurfil, P., "IFAS: Interactive Flexible Ad-Hoc Simulator", *Simulation Methods Practice and Theory*, Vol. 15, No. 7, August 2007, pp. 817-830.

[11] Calise, A. J., and Preston, D., "Swarming/Flocking and Collision Avoidance for Mass Airdrop of Autonomous Guided Parafoils", Proceedings of the AIAA Guidance, Navigation and Control Conference, San Francisco, CA, August 2005, Paper AIAA-2005-6477.

[12] Benolol, S., and Zapirain , F., "The FASTWing Project - Parafoil Development and Manufacturing" 18th AIAA Aerodynamic Decelerator Systems Technology Conference and Seminar, Munich, Germany, May, 2005, Madrid, Spain.

[13] Phillips, W. F., *Mechanics of Flight*, Wiley, New York, 2004.

[14] Stengel, R. F., *Optimal Control and Estimation*, Dover, New York, 1994.

[15] http://standards.ieee.org/getieee802/802.11.html