Coordination and Communication of Cooperative Parafoils for Humanitarian Aid

PINI GURFIL
SHARONI FELDMAN
MORAN FELDMAN
Technion–Israel Institute of Technology

In the case of a wide-scale disaster, an accurate airdrop of emergency supplies is crucial. 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 the 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 of the low complexity of the swap and replace recovery algorithms enable these procedures to run in a distributed environment under real-time limitations.

Manuscript received December 31, 2007; revised November 10, 2008 and March 29, 2009; released for publication April 23, 2009.

IEEE Log No. T-AES/46/4/XXXXX.

Refereeing of this contribution was handled by M-J. Tahk.

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.

Authors' addresses: P. Gurfil and S. Feldman, Faculty of Aerospace and Engineering, Technion–Israel Institute of Technology, Haifa 32000, Israel, E-mail: (pgurfil@technion.ac.il); M. Feldman, Faculty of Computer Science, Technion–Israel Institute of Technology, Haifa 32000, Israel.

I. INTRODUCTION

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

The ability of multiagent systems to meet complex mission requirements in an arbitrary theater under partial and uncertain information paves the way for applying multiagent methodologies to a variety of platforms in the air, on the ground, and in the 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 [7–9] and task assignment [10] methods. In particular [11] discusses UAV autonomy based on vision-aided navigation, and [12] developes a path-planning algorithm for UAVs that fly in urban environments.

While the literature on cooperative UAVs is abundant, the reported research efforts dedicated to developing cooperative parafoils are few [15]. Autonomous parafoils are equipped with sensors (typically GPS receivers and occasionally an inertial measurement unit) and with actuators that enable autonomous operation and precision landing at designated sites (see e.g. the parafoil developed in the FastWing project [16]). This ability is critical in times of emergency, such as in 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 preplanned 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 predesignated 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), which allows 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 can possibly cause a 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 1) 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 2) increases the chances for successfully completing a single airdrop task by creating redundancy based on judicious task management, without adding redundant parafoils.

II. 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.

A. 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. By denoting the wind velocity components in the NED coordinates by w_x and w_y (we assume that there is no wind shear), the equations of motion assume the form [14]

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

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

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

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

In (4) ψ is defined as the heading angle in wind axes, that is,

$$\tan\psi = \frac{\dot{y} - w_y}{\dot{x} - w_x}. \tag{5}$$

And φ denotes the bank angle, which, at steady state, assumes a constant value φ_d determined by a servo deflection δ. Thus the closed-loop bank angle dynamics are given by

$$\dot{\varphi}(t) = [-\varphi(t) + \varphi_d(\delta)]/\tau \tag{6}$$

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

The constant flight path angle γ that appears in (1)–(3) satisfies the relationship [17]

$$\tan\gamma = -\frac{C_D}{C_L\cos\varphi_d} \tag{7}$$

where C_D and C_L are the drag and lift coefficient, respectively.[1] Thus the dynamical model (1)–(6) includes two control variables: γ and φ_d (the servo deflection enables control of the lift and drag coefficients). Hence based on (4), at steady state,

$$\dot{\psi} = g\tan\varphi_d/V. \tag{8}$$

B. 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

$$x(t_f) = x_f, \qquad y(t_f) = y_f. \tag{9}$$

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

$$\dot{X} = \dot{x} - w_x(t), \qquad \dot{Y} = \dot{y} - w_y(t). \tag{10}$$

The resulting position components are computed by integrating (10):

$$X(t) = x(t) - \int_{t_0}^{t} w_x(\tau)d\tau, \qquad Y(t) = y(t) - \int_{t_0}^{t} w_y(\tau)d\tau. \tag{11}$$

This transforms (1) and (2) into

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

$$\dot{Y}(t) = V\cos\gamma\sin\psi(t). \tag{13}$$

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

$$X_0 = x_0, \qquad Y_0 = y_0 \tag{14}$$

and

$$X_f = x_f - \int_{t_0}^{t_f} w_x(\tau)d\tau, \qquad Y_f = y_f - \int_{t_0}^{t_f} w_y(\tau)d\tau \tag{15}$$

[1] A complete aerodynamic model must incorporate limits for the lift coefficient. Thus the gliding angle is actually constrained. However this fact is of little relevance for the subsequent discussion.

Mt2 □ job no. 2147 □ ieee □ aerospace and electronic systems □ 2147D12 [2] □ (XXX) □ 07-22-10 □ 12:36 PM

TABLE I
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

Case #	x_f [m]	y_f [m]	γ^* [rad]	φ_d^* [rad]	t_f [sec]	Δr_f [m]
1	−1000	−500	−0.7416	−0.00576	370.1	4.25
2	−1000	0	−1.0232	−0.00472	292.8	2.54
3	−1000	500	−1.1298	−0.00167	276.4	2
4	−1000	2000	−1	0.005	297	2.5
5	−500	1000	−0.9788	−0.001467	301.2	2.9
6	0	500	−0.7439	−0.00378	369.2	4.5
7	0	1000	−0.8223	−0.00233	341.1	3.97
8	1000	1000	−0.5129	−0.00291	509.4	7.28
9	1000	2000	−0.6194	−0.0014	430.6	6
10	4000	4000	−0.2834	−0.001299	893.9	14.3

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

$$t_f = -\frac{h_0}{V \sin\gamma^*}. \tag{16}$$

It is now required to find two equations for γ^* and φ_d^*. To this 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 $\varphi \approx \varphi_d^*$, and (cf. (8))

$$\psi(t) = \psi_0 + \frac{g\tan\varphi_d^*}{V}t. \tag{17}$$

To complete the procedure we integrate (12) and (13) with $\psi(t)$, as in (17), and substitute $t = t_f$. This yields the desired equations for γ^* and φ_d^*:

$$X_f(\gamma^*) = X_0 + \frac{V^2}{g\tan\varphi_d^*}\cos\gamma^*\left[\sin\left(\psi_0 + \frac{gt\tan\varphi_d^*}{V}\right) - \sin\psi_0\right] \tag{18}$$

$$Y_f(\gamma^*) = Y_0 + \frac{V^2}{g\tan\varphi_d^*}\cos\gamma^*\left[\cos\left(\psi_0 + \frac{gt\tan\varphi_d^*}{V}\right) + \cos\psi_0\right] \tag{19}$$

where X_f and Y_f depend on γ^* through (15). Equations (18) and (19) render two algebraic equations for γ^* and φ_d^*. If a solution exists then, at t_f, (9) is satisfied, and the parafoil reaches the target location with a miss distance induced by the servo time constant. This miss distance is defined by

$$\Delta r_f = \sqrt{[x(t_f) - x_f]^2 + [y(t_f) - y_f]^2}. \tag{20}$$

The required bank angle is achieved by a servo deflection, while the flight path angle is determined by changing the parafoil's drag-to-lift ratio (cf. (7)).

Sample results of the above process are given in Table I. 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/s, $w_y = 10$ m/s, and the initial conditions $X_0 = -1000$ m, $Y_0 = -2000$ m, $H_0 = 3000$ m. The servo time constant is $\tau = 0.01$ s. It is seen that the trajectory design algorithm provides miss distances ranging from 2 m to 14.3 m, depending on the final coordinates.

The reference trajectories of the parafoils are shown in Fig. 1. The initial position is designated by an "o," and the final position is designated by an "x." It is important to note that, in some scenarios, the predefined landing position cannot be accomplished under the given initial altitude and wind conditions; for example the coordinates (1000,500) yield a miss distance of 188 m. This highlights the need for a task assignment algorithm that is capable of changing the predesignated landing position in real time. This is the subject of Section III.

C. Guidance Algorithm

The purpose of the guidance algorithm is to steer the parafoils to the predetermined reference trajectories (as determined by the initial flight path angle and the steady-state bank angle) in the presence of initial release errors and of perturbations, which cause the actual trajectory to deviate from the reference trajectory. Let $X(t)$, $Y(t)$, and $h(t)$ 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

$$\begin{aligned} \delta X(t) &\triangleq X(t) - X^*(t), \qquad \delta Y(t) \triangleq Y(t) - Y^*(t) \\ \delta h(t) &\triangleq h(t) - h^*(t), \qquad \delta\psi(t) \triangleq \psi(t) - \psi^*(t). \end{aligned} \tag{21}$$

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

$$\delta\gamma(t) \triangleq \gamma(t) - \gamma^*, \qquad \delta\varphi(t) \triangleq \varphi(t) - \varphi^* \tag{22}$$

become control inputs. The trajectory control is performed in closed loop by 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

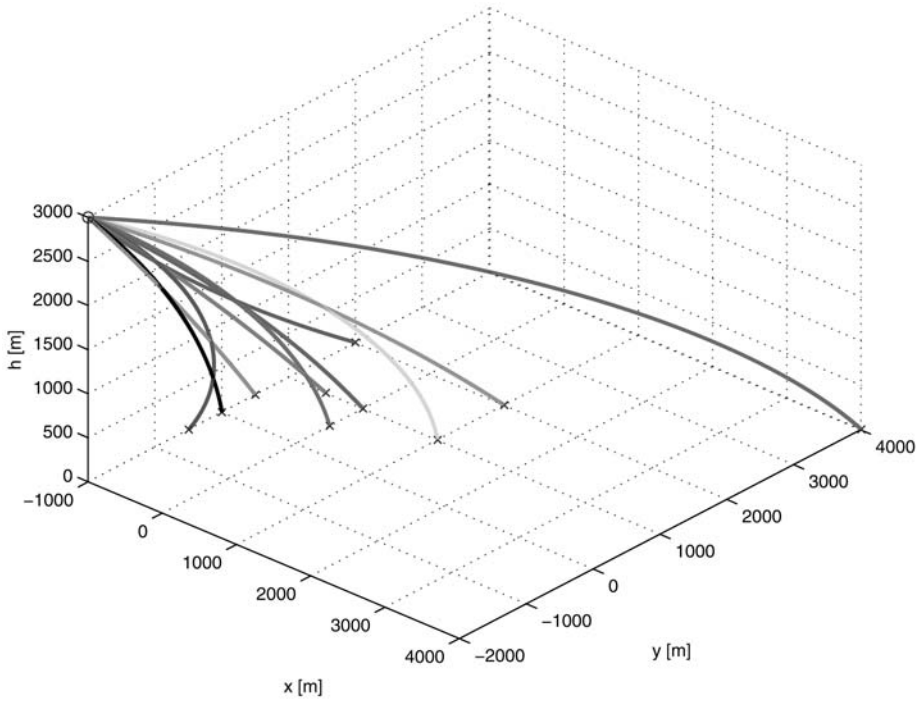


Fig. 1. Sample parafoil reference trajectories in NED coordinates.

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 (3), (12), and (13) about the nominal trajectory yields the linear time-varying model

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

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

$$A(t) = \begin{bmatrix} 0 & 0 & -V\cos\gamma^*\sin\psi^*(t) \\ 0 & 0 & V\cos\gamma^*\cos\psi^*(t) \\ 0 & 0 & 0 \end{bmatrix}$$
$$B(t) = \begin{bmatrix} -V\sin\gamma^*\cos\psi^*(t) & 0 \\ -V\sin\gamma^*\sin\psi^*(t) & 0 \\ 0 & \dfrac{g}{V}(1+\tan^2\varphi^*) \end{bmatrix}. \tag{24}$$

A straightforward approach for designing a closed-loop trajectory controller for system is neighboring optimal control [18].

Given a cost functional of the form

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

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 is given by

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

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

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

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

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

To illustrate the performance of the said guidance law, we perform a simulation of the nominal trajectory described by case 3 in Table I 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 Fig. 2, which depict 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, which nullifies the altitude difference at impact so that the miss distance is identical to the value given in Table I.

III. 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

Mt2 □ job no. 2147 □ ieee □ aerospace and electronic systems □ 2147D12 [4] □ (XXX) □ 07-22-10 □ 12:36 PM

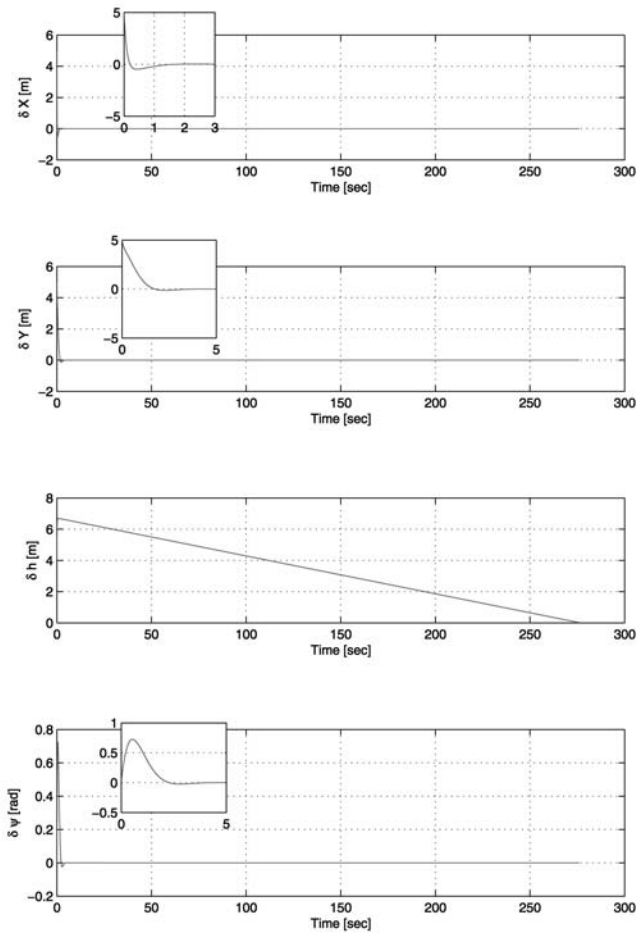


Fig. 2. Guidance using neighboring optimal trajectory control. After transient of about 2 s, X and Y components converge to zero, while altitude difference is nullified at impact.

mechanism for target reassignment. To illustrate the importance of task management, we begin with a simple motivating example.

A. 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 predefined allowed miss distance) is p. The target of parafoil P_h has the highest priority, and the remaining $N-1$ parafoils have targets with lower priorities. We are interested in evaluating the probability that the target of P_h will get the needed supply. To this 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. Let n be the number of parafoils that survive the search. The probability distribution of n is given by the familiar binomial distribution

$$p(n=k) = \binom{n}{k} p^k (1-p)^{n-k}, \qquad 0 \le k \le N. \tag{29}$$

The probability of success in attending target P_h, denoted by $\Pr(P_h)$, is then

$$\Pr(P_h) = \sum_{k=0}^{N-1} \binom{N}{k} p^k (1-p)^{N-k}. \tag{30}$$

Thus if $N=2$, then the probability that the parafoil P_h will reach 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 0.99. Similarly if $N=3$, then $\Pr(P_h)=0.999$.

This simple example highlights the importance of task management, which is to be discussed in the remainder of this section.

B. 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 are concentrated 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 develop an interactive design tool (IDT). Fig. 3 presents a screenshot of the IDT. The OM first identifies the coordinates of the 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 to determine the parafoil release coordinates. Release coordinate are defined as the center of an airdrop envelop that covers a given set of thumbtacks, as shown in Fig. 3. 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 (18)–(19) given $h(t_0)$.

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

C. 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.

Mt2 □ job no. 2147 □ ieee □ aerospace and electronic systems □ 2147D12 [5] □ (XXX) □ 07-22-10 □ 12:36 PM

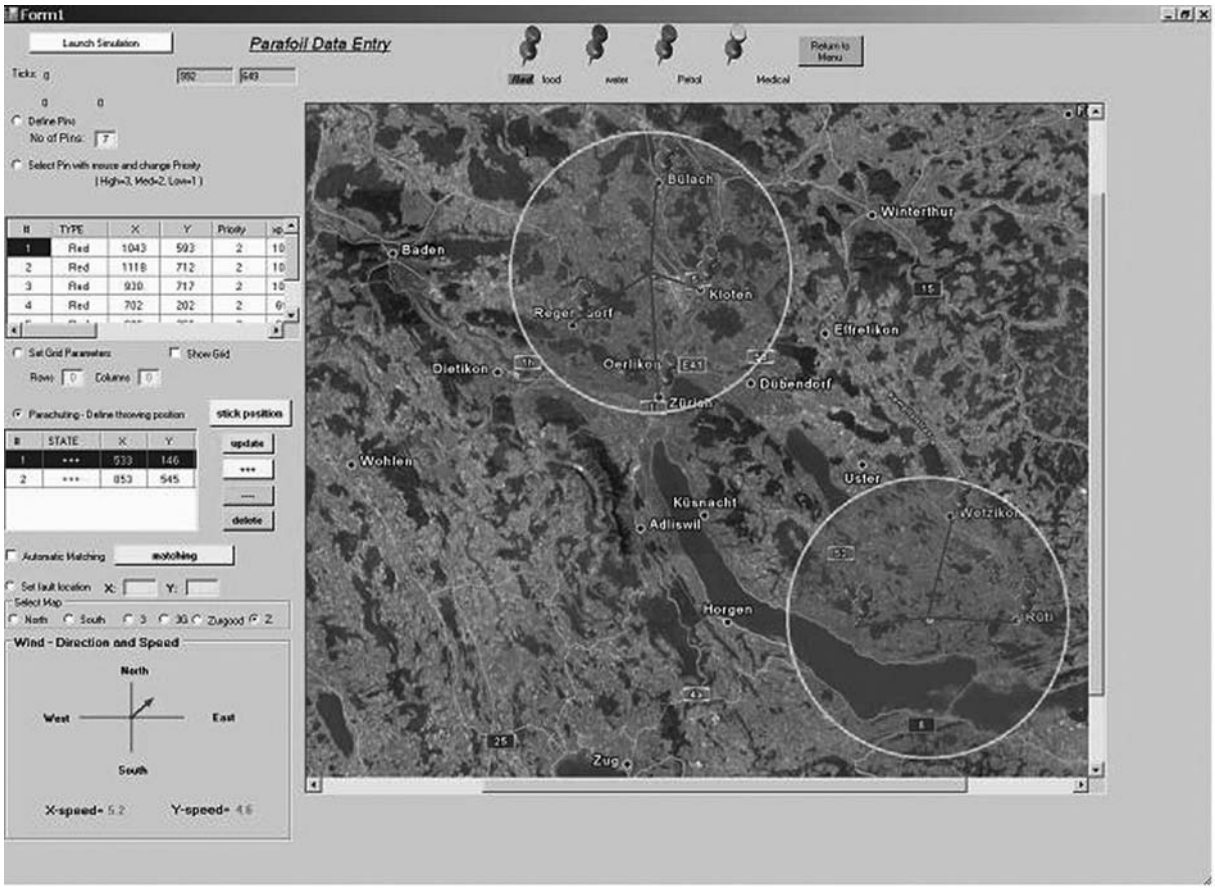


Fig. 3. Snapshot of mission planning screen. Interface provides interactive design tool for prioritizing target locations in need of various support types and calculating airdrop envelopes.

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 where 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 eventhough 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 Fig. 4.

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 commences on a recovery procedure. This procedure starts with a Data Collection step (Fig. 4), 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 (Fig. 4). 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

$$J = \sum_i \Pi_i \tag{31}$$

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

$$J^* = \max J_G. \tag{32}$$

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

The Glide state, shown in Fig. 4, constantly monitors the trajectory by comparing precalculated reference points on the reference trajectory to the

Mt2 □ job no. 2147 □ ieee □ aerospace and electronic systems □ 2147D12 [6] □ (XXX) □ 07-22-10 □ 12:36 PM

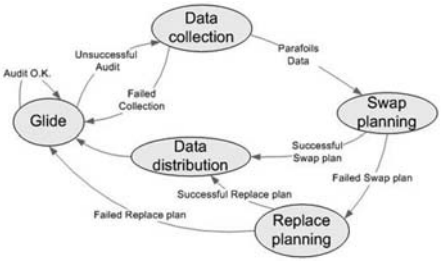


Fig. 4. Task management state diagram.

TABLE II
Basic Swap and Replace Rules

	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	

actual trajectory. The Glide process initiates corrective maneuvers according to (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 declares the parafoil as faulty. Another event that leads to a declaration of a parafoil as faulty is when Glide gets a malfunction indication from one of the hardware elements. The type and the severity of the fault are major factors in deciding whether a parafoil will be able to participate in either Swap or Replace, which in turn depends 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). By using the ad-hoc communication infrastructure, to be described shortly, a faulty parafoil collects data from the surrounding parafoils and calculates J^* by using the breadth-first search algorithm described in the Appendix.

D. Swap and Replace Decision Logic

Table II 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).

E. Swap and Replace Illustrative Examples

In this section we illustrate Swap and Replace by using a few simple examples. To begin Fig. 5

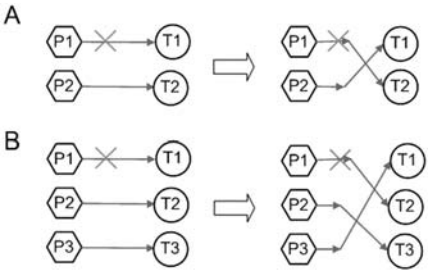


Fig. 5. Schematic description of 2-way and 3-way parafoil Swap processes. In each case, targets get their required supplies, but not by original parafoils assigned to those targets.

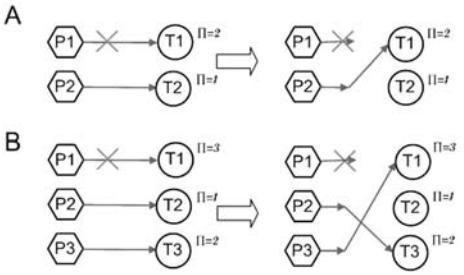


Fig. 6. Schematic description of 2-way and 3-way parafoil Replace. This process is responsible for reassignment of targets in case of parafoil failure that cannot be remedied using Swap.

depicts a 2-way and a 3-way parafoil Swap scenarios. In Fig. 5, 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.

In Fig. 5, 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.

To illustrate the Replace procedure, Fig. 6 presents 2 scenarios. In Fig. 6, A presents a simple case. P_1 is unable to complete its mission, and Swap is 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.

In Fig. 6, B presents a more complicated case wherein a 3-way parafoil Replace is required.

In the example presented in Fig. 7, an arc $P_x \rightarrow P_y$ indicates that P_y can replace P_x. After analyzing this PSG P_f determines its ability to be replaced by another parafoil or, preferably, to swap targets with another parafoil.

In Fig. 7, A presents the decision-making process after P_f declares itself a faulty parafoil and completes building the PSG. The following replacements are applicable.

Mt2 □ job no. 2147 □ ieee □ aerospace and electronic systems □ 2147D12 [7] □ (XXX) □ 07-22-10 □ 12:36 PM

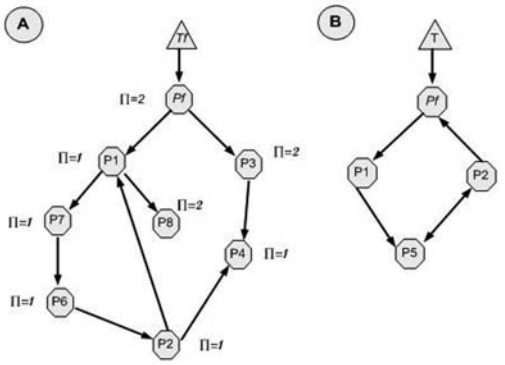


Fig. 7. Potential Swap graph examples.

1) P_1 replaces P_f as $\Pi(T_1) < \Pi(T_f)$. In this case T_1 is abandoned.

2) The second is 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 is abandoned.

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

In Fig. 7, 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 is initiated. So P_1 replaces P_f, P_5 replaces P_1, P_2 replaces P_5, and P_f replaces P_2. Note that, in this case, the priority is irrelevant.

IV. COMMUNICATION

The 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 predesignated targets as accurately as possible.

A. 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 of 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) that are network members.

1) *Radio Propagation Model*: 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 and assumes an RF propagation that uses a free-space transmission between two points at a distance d, is given by

$$\text{loss} = 92.5 + 20 \times \log(d \times f) \tag{33}$$

where d is in units of kilometers and where 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

$$\text{loss} = 40 \times \log(d) - 20 \times \log(Ht \times Hr) \tag{34}$$

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

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

B. Metrical Routing Algorithm

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

The MRA attempts to connect the parafoil group $G = \{P_1, P_2, \ldots, P_N\}$ by a minimal set of rooted trees that preserve geographical distances; viz. distances on the rooted trees are usually proportional to the distances of $P_1, P_2, \ldots, P_N$ 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 attempts to cover $G(t)$ by a minimal set of spanning trees. The rooted trees created by the MRA can be naturally used for both distributed computing (of, e.g., the applicative layer) as well as for communication and for data propagation tasks.

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 a demanding traffic load successfully 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 the transmission range of any node in the other trees, the trees function autonomously. As soon as a radio connection is created between two nodes, the trees are 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 a need to update the trees. A primary goal of the algorithm is to identify these changes and to adapt the tree structure to the new state. In the following discussion we 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 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.

Fig. 8 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). 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 Fig. 9(a) and (b). 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 initiates a merge process that results in a single tree.

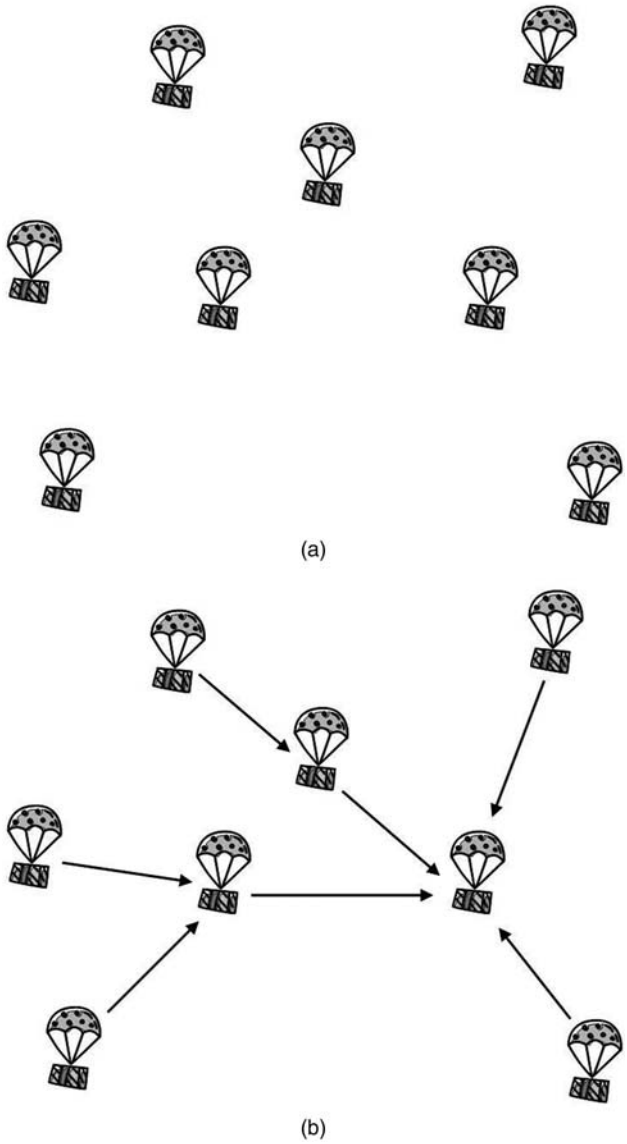
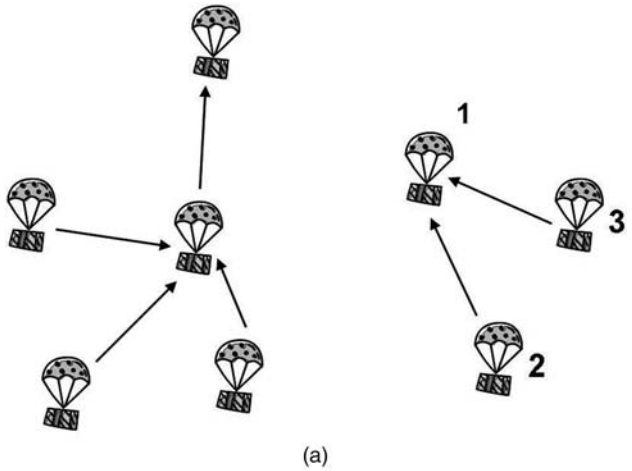


Fig. 8. Tree formation process.

C. Implementing Replace and Swap

The same AL package runs in all parafoils, and the following types of messages are exchanged.

1) A direct message, which is sent from the source node to a target. Sending a direct message implies that



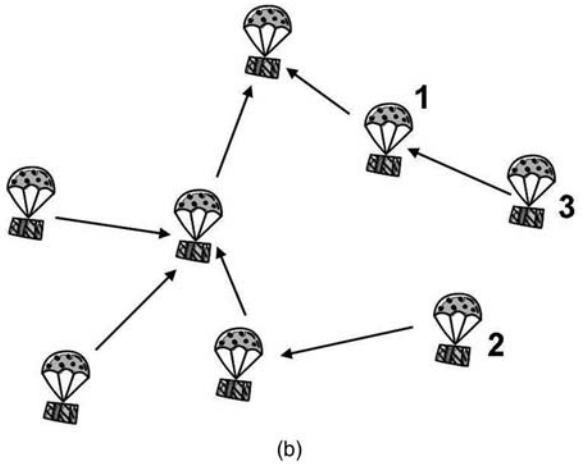


Fig. 9. Fusion of trees.

the source node is 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 of the number of nodes that create the network.

Regardless of the message type, the nodes are unaware of the path that a message passes through 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.

Fig. 10 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, its priority, and its own PID. Every parafoil that receives this message

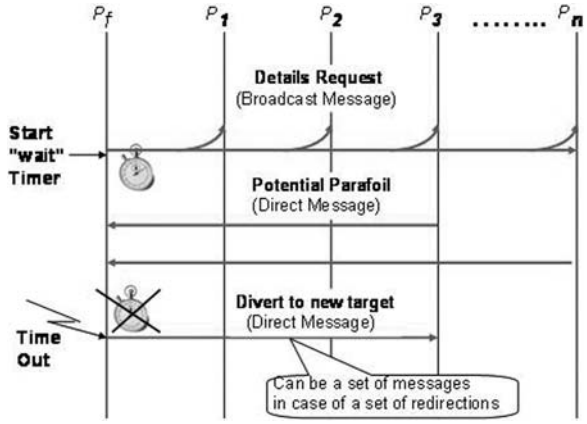


Fig. 10. Alternate parafoil selection process.

performs the following actions 1) it continuous the broadcast process by sending the message to its parent and offspring in the tree (it does not send the message to the sender to prevent loops) and 2) 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 if its own task differs the faulty parafoil's task, it discards the message (P_2 in Fig. 10). If its priority is lower than the originator's priority and if its task is identical to the originator's task, then it returns a direct message to the initiating parafoil (P_3 in Fig. 10).

In the direct message returned to the originator, the parafoil indicates its target priority, its location, and its own PID. The originating parafoil collects the answer messages that arrive 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 that it should start analyzing the answers immediately. The analysis of the answers is aimed at selecting the most suitable parafoil to replace the faulty parafoil. 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.

D. 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 is able to start Swap or Replace immediately. The idle load on every node consists of the inbound traffic (load on the receiver) and the outbound traffic (load on the transmitter). Table III presents the messages and the frequency of sending and receiving the idle load messages. The recovery from a fault requires the transmission of extra messages in a very short time. This traffic (presented in Fig. 10) consists of the following messages.

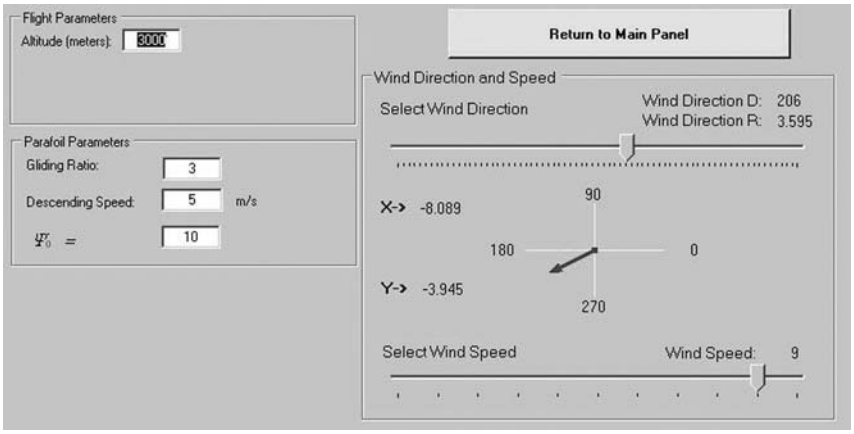


Fig. 11. Wind direction and speed definition.

TABLE III
Idle Time Messages and Frequency of 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

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 it is generated mostly by P_f and less on the parafoils that participate in the search for a replacing parafoil. Table IV presents the maximal calculated load in bytes/s during idle time and recovery time. The given numbers represent very moderate requirements that can be satisfied by any basic radio equipment.

V. 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.

A. 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 has been expanded to support cooperative parafoils simulation.

TABLE IV
Maximal Inbound and Outbound Traffic Load (bytes/s)

	Idle Time Traffic	Recovery Time—Extra Traffic	Total
Inbound	1800	800	2600
Outbound	170	800	970

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 online analysis, and it hosts a set of tools that can be used on the output log files created during the run. An important element of 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. Fig. 11 presents, for example, a part of a window used to graphically define the wind speed and direction. Fig. 12 shows some snapshots from the main simulation screen.

B. The Scenario

In Fig. 12 we present a scenario of the design, analysis, glide, and recovery steps. In this scenario we drop 5 parafoils that are carrying identical payloads required in 5 different locations. In Fig. 12, A presents the first phase of the airdrop process.

The OM defines the airdrop position by fixing thumbnails on the target positions. The number near every thumbtack represents the target priority (the

Mt2 □ job no. 2147 □ ieee □ aerospace and electronic systems □ 2147D12 [11] □ (XXX) □ 07-22-10 □ 12:36 PM

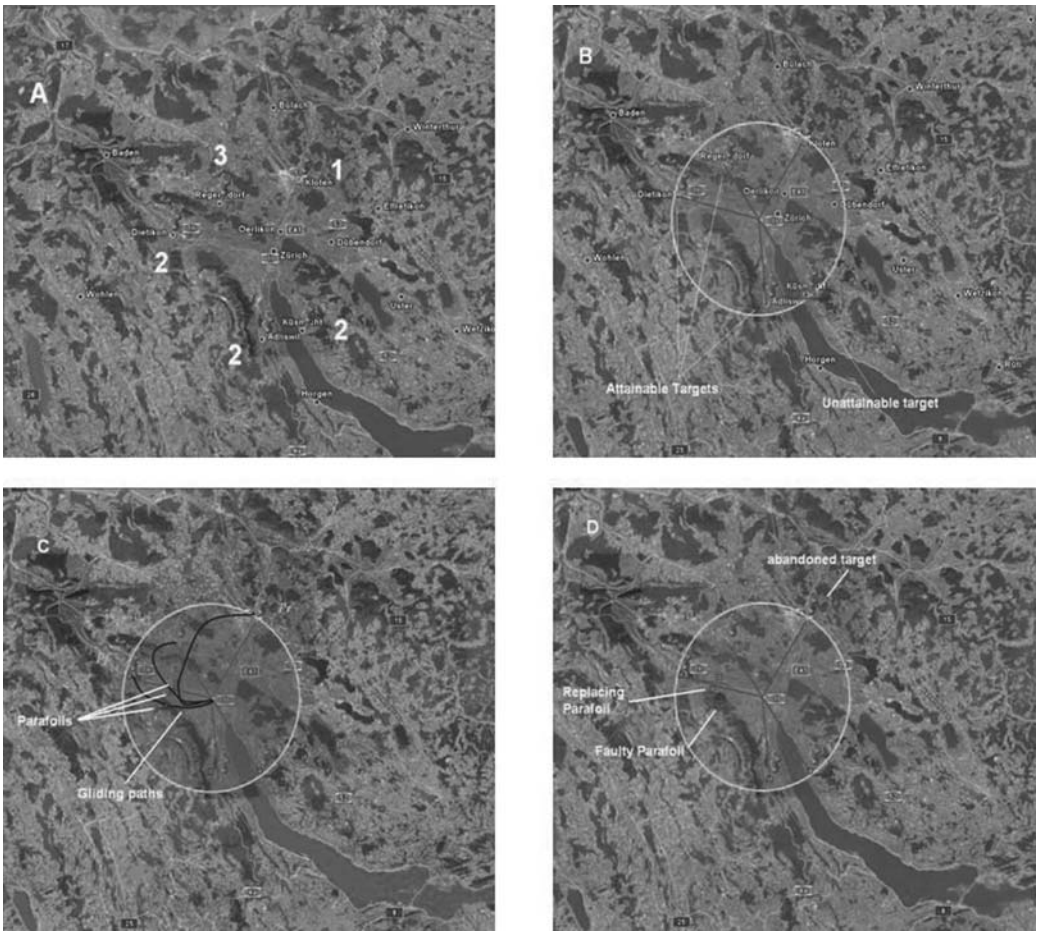


Fig. 12. Snapshot of airdrop and parafoil recovery scenarios.

default value is 2). In our example 3 targets have priority 2, while the other 2 targets have priorities 1 and 3.

In Fig. 12, B presents the next step of the dropping design process. The operator marks the preferred airdrop position, and the simulator analizes the ability of every parafoil to land at its target location.

In Fig. 12, 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 that it was added to the screenshot. An uninterrupted gliding process ends when the parafoils reach their targets.

In Fig. 12, D presents a case where one of the parafoils fails. We manually define a fault position on the simulator screen. 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 the target with the minimal priority.

Fig. 13 depicts the 3D trajectories of the failing parafoil P_f and of the replacing parafoil P_r, which were initially targeted to T_f and T_r, respectively. (P_f and P_r are also presented in C of Fig. 12.) 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 m. P_f fails after 472 s at an altitude of $h = 1612$ m. It initiates the Replace process. As the priority of T_f is higher than the priority of T_r and as P_r can replace P_f, P_r diverts its original trajectory at an altitude of 1808 m to an alternative trajectory that terminates at T_f after 1170 s.

VI. CONCLUSIONS

In this paper we present an innovative approach for handling a humanitarian aid mission based on a group of autonomous cooperative parafoils. The approach combines a guidance algorithm, an interactive planning tool used to plan the mission, and a task management logic that upgrades the parafoils from individual entities to a cooperative system that

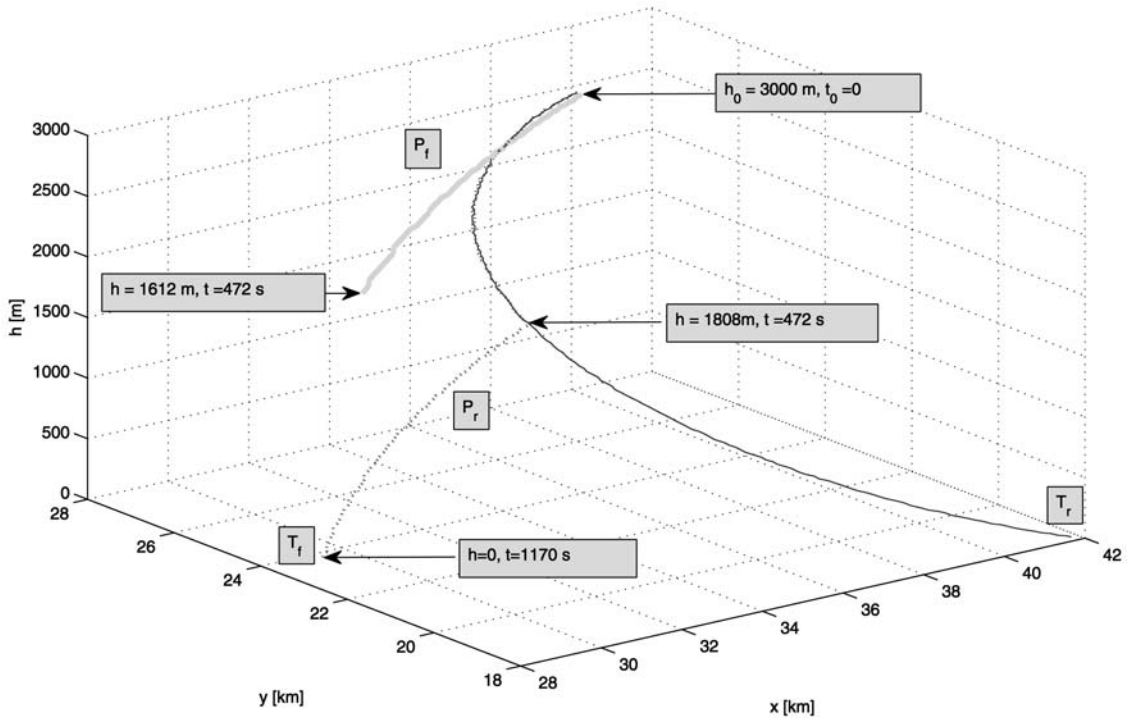


Fig. 13. Gliding trajectories of the replacing (P_r) and failing (P_f) parafoils. Replacing parafoil engages original target of failing parafoil.

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 is useful in humanitarian airdrop missions.

APPENDIX. THE BREADTH-FIRST SEARCH ALGORITHM

The breadth-first search (BFS) algorithm is an uninformed search method that systematically aims to expand and examine all nodes of a graph in search of a solution. It exhaustively searches the entire graph without an a priori consideration of the goal. The complexity of the BFS algorithm, in our case, is linear $O(|V| + |E|)$, where V is the number of parafoils and where E is the number of edges in the PSG graph. 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
```

REFERENCES

[1] Dudek, G., Jenkin, M., Milios, E., and Wilkes, D.
A taxonomy for multiagent robotics.
Autonomous Robots, **3** (1996), 375–397.

[2] Reif, J. and Wang, H.
Social potential fields: A distributed behavioral control for autonomous robots.
Robotics and Autonomous Systems, **27** (1999), 171–194.

Mt2 □ job no. 2147 □ ieee □ aerospace and electronic systems □ 2147D12 [13] □ (XXX) □ 07-22-10 □ 12:36 PM

[3] Mataric, M. J.
Behavior based control: Examples from navigation, learning, and group behavior.
Journal of Experimental and Theoretical Artificial Intelligence, **9**, 2–3 (Apr. 1997), 323–336.

[4] Reynolds, C. W.
Flocks, herds, and schools: A distributed behavioral model.
Computer Graphics, **21** (1987), 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.
In *Proceedings of the Cooperative Control Workshop*, Florida, Dec. 2000.

[6] Parunak, H. V. D., Purcell, M., and O'Connell, R.
Digital pheromones for autonomous coordination of swarming UAVs.
In *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.
In *Proceedings of the IEEE International Conference on Robotics and Automation*, Seoul, South Korea, May 2001.

[8] Beard, R. W., McLain, T. W., Goodrich, M. A., and Anderson, E. P.
Coordinated target assignment and intercept for unmanned air vehicles.
IEEE Transactions on Robotics and Automation, **18**, 6 (Dec. 2002), 911–922.

[9] Yang, G. and Kapila, V.
Optimal path planning for unmanned air vehicles with kinematic and tactical constraints.
In *Proceedings of the 41st IEEE Conference on Decision and Control*, vol. 2, Dec. 2002, 1301–1306.

[10] Rasmussen, S., Chandler, P., Mitchell, J. W., Schumacher, C., and Sparks, A.
Optimal vs. heuristic assignment of cooperative autonomous unmanned air vehicles.
In *Proceedings of the AIAA Guidance, Navigation & Control Conference*, Austin, TX, 2003.

[11] Bryson, M. and Sukkarieh, S.
Observability analysis and active control for airborne SLAM.
IEEE Transactions on Aerospace and Electronic Systems, **44**, 1 (Jan. 2008), 261–280.

[12] Shima, T., Rasmussen, S., and Gross, D.
Assigning micro UAVs to task tours in an urban terrain.
IEEE Transactions on Control Systems Technology, **15**, 4 (July 2007), 601–612.

[13] Ben-Asher, Y., Feldman, M., and Feldman, S.
Ad-hoc routing using virtual coordinates based on rooted trees.
In *Proceedings of the IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing* (SUTC 2006), Taichung, Taiwan, July 2006.

[14] Ben-Asher, Y., Feldman, M., Feldman, S., and Gurfil, P.
IFAS: Interactive flexible ad-hoc simulator.
Simulation Methods Practice and Theory, **15**, 7 (Aug. 2007), 817–830.

[15] Calise, A. J. and Preston, D.
Swarming/flocking and collision avoidance for mass airdrop of autonomous guided parafoils.
Journal of Guidance, Control and Dynamics, **31**, 4 (July–Aug. 2008).

[16] Benolol, S. and Zapirain, F.
The FASTWing project—Parafoil development and manufacturing.
Presented at 18th AIAA Aerodynamic Decelerator Systems Technology Conference and Seminar, Munich, Germany, May 2005.

[17] Phillips, W. F.
Mechanics of Flight.
Hoboken, NJ: Wiley, 2004.

[18] Stengel, R. F.
Optimal Control and Estimation.
New York: Dover, 1994.

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

Pini Gurfil received his Ph.D. in aerospace engineering from the Technion–Israel Institute of Technology, in 2000.

From 2000 to 2003 he was with Princeton University's Department of Mechanical and Aerospace Engineering, where he served as a research staff member and lecturer. In September 2003 he joined the faculty of Aerospace Engineering at the Technion. He is the founder and director of the Distributed Space Systems Laboratory at the Technion. He has been conducting research in astrodynamics, distributed space systems, vision-based navigation, optimization and multiagent systems.

Dr. Gurfil has published over 140 journal and conference articles in the areas of his research interests.

Sharoni Feldman received his B.Sc. degree in computer science from the Technion–Israel Institute of Technology, his M.Sc. degree in business administration in the field of information systems from Tel Aviv University, and his Ph.D. in computer science from Haifa University, Israel.

From 2006 to 2008 he served as a research staff member on the faculty of aerospace engineering at the Technion. He is a consultant to Israeli Aerospace Industries (IAI) and other high-tech companies.

Moran Feldman received his B.A. and M.Sc. (Magna Cum Laude) degrees in computer science from the Israeli Open University in 2004 and 2008, respectively.

Currently he is a Ph.D. student in the Department of Computer Science at the Technion–Israel Institute of Technology.