Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Control Of An Airship Using Particle Swarm Optimization and Neural Network

Ruting Jia[1], Michael T. Frye[2], Chunjiang Qian[1]
[1]Electrical and Computer Engineering Department, University of Texas at San Antonio, San Antonio, TX 78249, USA
Ruting.jia@utsa.edu
[2]Engineering Department, University of the Incarnate Word, San Antonio, TX 78249, USA
mfrye@uiwtx.edu

***Abstract*—The objective of this paper is to design an optimized controller for the Tri-turbofan Airship model. In lieu of using the traditional controller analysis method, the Particle Swarm Optimization algorithm for controller optimization has been implemented. For more accurate results, this research used an updated neural network model to approximate the actual Tri-turbofan Airship dynamics. The effectiveness of the PSO algorithm will be shown by the simulation in an updated neural network model, compared to a linear model of the Tri-turbofan model.**

***Keywords*—Particle swarm optimization, dynamic neural network model, real time optimal control**

I. Introduction

A particularly revealing issue caused by severe weather such as Hurricane Katrina was the difficulty of establishing communication among survivors, first responders, and the coordinating agencies in the areas where both cell phone and landline telephone are heavily damaged. Furthermore, the difficulty of surveillance of surface topography under difficult weather requires the need for a system that can operate in areas that might be dangerous for manned vehicles. Both rotorcrafts and fixed-wing aircrafts are utilized in aerial tasks whenever high altitude is involved but can have expenses and short loiter times. Under some circumstances, the use of airships to perform the work of surveillance and or as a temporary cell phone network can be more economical, efficient, and safer.

In this paper, we intend to show that a real time optimal control method can be used to achieved stable flight conditions for the movement of an airship under the constraints of minimal energy usage, limited control actuation, and set point requirements; all the while minimizing the effects of environmental disturbances. The requirements for the optimal control are:

1. Real-time convergence to a solution
2. Minimal control energy
3. Optimal Trajectory Solution

In order to meet these criteria, a real-time optimal trajectory generation method has been developed in conjunction with a feedback controller which can accommodate constraints.

This paper examines a method to design an optimized controller for the Tri-turbofan Airship model developed in [1]. In order to get better performance for real time optimal control of an airship, we need a fast, stable, reliable method to update the controller. Instead of using the traditional approach, this paper explores using the Particle Swarm Optimization (PSO) algorithm for an optimized solution of the controller.

The PSO algorithm is a stochastic, population-based evolutionary computation technique developed by Dr. Russell C. Eberhart and Dr. James Kennedy in 1995 [2]. It has been shown to be effective in optimizing difficult multidimensional problems in many fields. PSO has a fixed simple and fast algorithm which is similar to an evolutionary algorithm but simpler and has less parameter needed to be adjusted.

In order to have a better representation of the real airship dynamics, we used large amount of observations of real airship flight data to get a neural network model and then keep updating the model with new data to achieve the real time dynamic neural network model.

This paper is organized as follows: Section II describes the linear, nonlinear and neural network model of the Tri-Turbofan airship used for surveillance airship control research. Section III describe the rules on how the Particle Swarm Optimization algorithm works and then reformulate the airship model to a PSO solvable problem to handle the control constraints. Section IV gives the simulation result of the PSO method applied to linear and neural network model. Section V concludes and summarized the paper.

II. Airship Model

The objective of this paper is to investigate the feasibility of using particle swarm optimization technique to solve the airship flight control in a fast, exact, and efficient manner. We are using both a 10 and a 20 [ft] airship for testing and verification of the control techniques. As part of the airship research, a tri-motor airship was used for developing an understanding of airship flight dynamics and for collecting flight test data. The airship used in is the Tri-Turbofan airship shown in Figure 1 [3].

Figure 1. Tri-Turbofan Remote-Controlled Airship.

A. Nonlinear Model

- u – longitude velocity
- v – lateral velocity
- w – vertical or normal velocity
- q – pitch rate
- p – roll rate
- r – raw rate

Due to the size of an airship's envelope volume, the displaced air's mass and inertia effects on the dynamics of an airship cannot be ignored as is the case in aircraft design [4,5].

The 6-DOF first principle force model described by the following equations of motion of an airship is based on [6] and roughly described *ma*=*F*,

$$
\begin{aligned}
& m_x\dot{u} + \left(ma_z - \dot{x}_{\dot{q}}\right)\dot{q} \\
& = -m_z qw + m_y rv + ma_z pr + ma_x\left(q^2 + r^2\right) + X \\
& m_y\dot{v} - \left(ma_z + \dot{Y}_{\dot{p}}\right)\dot{p} + \left(ma_y + \dot{Y}_{\dot{r}}\right)\dot{r} \\
& = -m_x ru + m_z pw - ma_x pq - ma_z qr + Y \\
& m_z\dot{w} - \left(ma_x - \dot{Z}_{\dot{q}}\right)\dot{q} \\
& = -m_y pv + m_x qu - ma_x pr + ma_z\left(q^2 + q^2\right) + Z \\
& J_x\dot{p} - \left(ma_z + \dot{L}_{\dot{v}}\right)\dot{v} - J_{xz}(\dot{r} + pq) \\
& = \left(J_y - J_z\right)qr + ma_z\left(ru - pw\right) + L \\
& J_y\dot{q} + \left(ma_z - \dot{M}_{\dot{u}}\right)\dot{u} - \left(ma_x + \dot{M}_{\dot{w}}\right)\dot{w} \\
& = -\left(J_x - J_z\right)pr - ma_z\left(qw - rv\right) + J_{xz}\left(r^2 - p^2\right) + \cdots \\
& + ma_x\left(pv - qu\right) + M \\
& J_z\dot{r} - J_{xz}\dot{p} + \left(ma_x - \dot{N}_{\dot{v}}\right)\dot{v} \\
& = \left(J_x - J_y\right)pq - J_{xz}pr - ma_x\left(ru - pw\right) + N
\end{aligned}
\tag{1}
$$

Where the apparent masses are defined by $m_x = m - \dot{X}_{\dot{u}}$, $m_y = m - \dot{Y}_{\dot{v}}$, and $m_z = m - \dot{Z}_{\dot{w}}$. The apparent moments of inertia are defined as $J_x = I_x - \dot{L}_{\dot{p}}$, $J_y = I_y - \dot{M}_{\dot{q}}$, and $J_z = I_z - \dot{N}_{\dot{r}}$. Finally the apparent products of inertia are $J_{xz} = I_{xz} + \dot{N}_{\dot{p}}$ and $J_{xy} = J_{yz} = 0$.

B. Linearized Model

For controller analysis, a linear longitudinal model of the Tri-Turbofan airship based on the nonlinear model of (1) was developed. Using the small perturbation method for linearization (refer to [7] and [6] for further details on the method), the products and squares become relatively small and can be ignored. Furthermore, since the airship attitudes θ; φ; ψ; are assumed to be small for flight control analysis, the cosine and sine angles are small and can be ignored. Finally, the cross coupling of the linear and angular rates between the longitudinal and lateral-directional axes can be ignored because of the relatively small attitude angles.

Applying the small perturbation method assumptions on the longitudinal axis of the Tri-Turbofan airship and adding the attitude angle θ, the nonlinear equations of motion from (1) reduce to the following linear equations:

$$
\begin{aligned}
& m_x\dot{u} + \left(ma_z - \dot{X}_{\dot{q}}\right)\dot{q} = X_a + X_b + X_g + X_p \\
& m_z\dot{\omega} - \left(ma_x - \dot{Z}_{\dot{q}}\right)\dot{q} = Z_a + Z_b + Z_g + Z_p \\
& J_y\dot{q} + \left(ma_z - \dot{M}_{\dot{u}}\right)\dot{u} - \left(ma_x + \dot{M}\dot{\omega}\right)\dot{\omega} = M_a + M_b + M_g + M_p \\
& \dot{\theta} = \mathrm{q}
\end{aligned}
\tag{2}
$$

C. Neural Network Model

Note that both the nonlinear and linearized models were derived under certain assumptions on the working conditions. The models might not be able to represent the real airship system when the working condition changes. In order to get a better approximation model of the Tri-Turbofan airship, we have developed a model by training through Neural Network using collected real flight data [8]. We use 150 groups of flight inputs and outputs to train the model. Figure 2 is the compare of real and Neural Network output.

From the following simulation result, we can see the neural network training model gives a good match output with the real flight test output. Parameters are defined as the longitudinal velocity u in [ft/s], vertical velocity w in [ft/s], angular pitch rate q in [deg/s], and theta θ in [deg], vertical position x in [m]. These parameters are same either as inputs or as outputs. By updating the neural network model through new coming data, the model will tend to be as close as real.

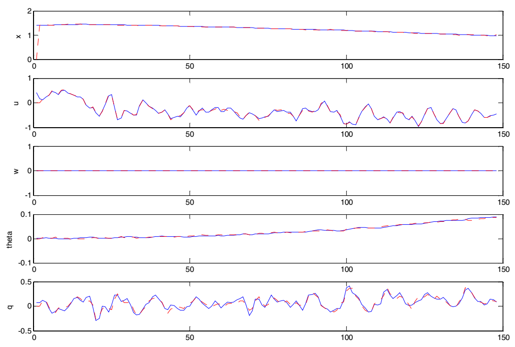

Figure 2. Compare of Real Output & Neural Network Output.

Figure 3 below is the decoupled nonlinear model on the longitudinal dimension. It uses 1 to 100 groups of data to train the neural network. On next section, we will explain how to find the optimized controller for the airship on the decoupled model.

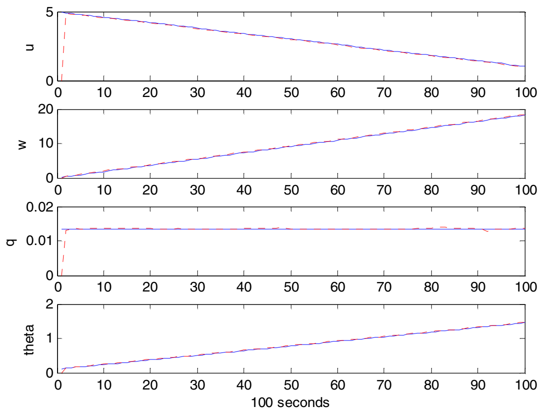

Figure 3. Neural Network Decoupled Model.

III. Particle Swarm Optimization based Controller

A. Particle Swarm Optimization Algorithm

Particle swarm optimization is a swarm intelligence based algorithm to find a solution to an optimization problem in a search space, or model and predict social behavior in the presence of objectives. The system initially randomizes a group of solutions, and then after several iterations finds the optimized solution. It doesn't have the crossover and mutation as the Genetic Algorithm (GA) do. Compare to GA, PSO is easy to implement and just a few parameters need to be adjusted.

The population is called a swarm and the individuals are called particles. Every particle moves in the search space with an adaptable velocity, and records the best position it ever encountered. Moreover the best position ever attained by all individuals of the swarm is communicated to all the particles. The swarm is manipulated according to the following equations:

$$V_n = \omega * V_n + C_1 * rand() * (p_{best,n} - X_n)$$
$$+ C_2 * rand() * (g_{best,n} - X_n) \quad (2)$$

$$X_n = X_n + \Delta t * V_n \quad (3)$$

Where V_n is the velocity of the particle in the nth dimension, X_n is the particle's position in the nth position, rand () is random numbers, uniformly distributed within the interval [0, 1] and C_1, C_2 are positive constants, called the acceleration constant. C_1 is a factor determining how much the particle is influenced by pbest, and C_2 is a factor determining how much the particle is influenced by gbest. ω is a parameter called the inertial weight.

The following Figure 4 is particles moving trend [9]. Individual particles (1&2) are accelerating toward the location of the global best solution gbest, and the location of their own personal best location pbest. Figure 4 shows how the personal best location and the global best location will affect one particle's next step's velocity. To determine the updated velocity is one of the core parts of PSO algorithm. When a particle is close enough to the optimal gbest, it should slow down so that the particle won't miss the gbest and go farther away. And on the other hand, if a particle is far away from the optimal gbest, it should maintain a high speed velocity to search the optimal solution.

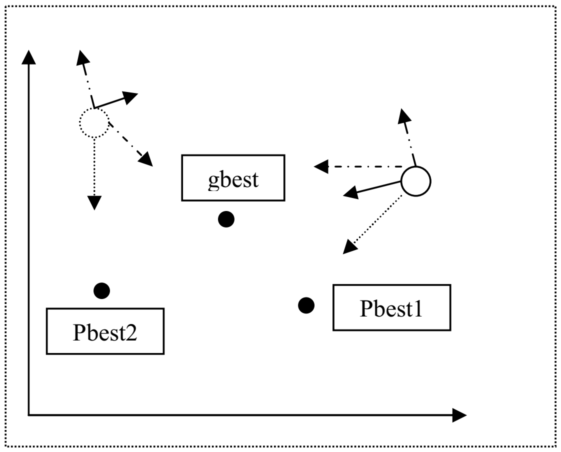

Figure 4. Particles Moving Trend

The first step to implement PSO is to define a solution space which is also called a searching space. If we give an optimal searching range, the PSO will quickly converge to an optimal solution. The second step is to define a fitness function to evaluate how good the solution is. For out problem, the definition of fitness function is given at Section IV. The third step is to initialize the swarm. The initialization of the swarm and the velocities is performed randomly in the search space. For the velocity, both the direction and value are randomly generated at the very beginning. And the swarm will update its best value every cycle based on the equations (2), (3). The PSO algorithm's flow chart shows in Figure 5.

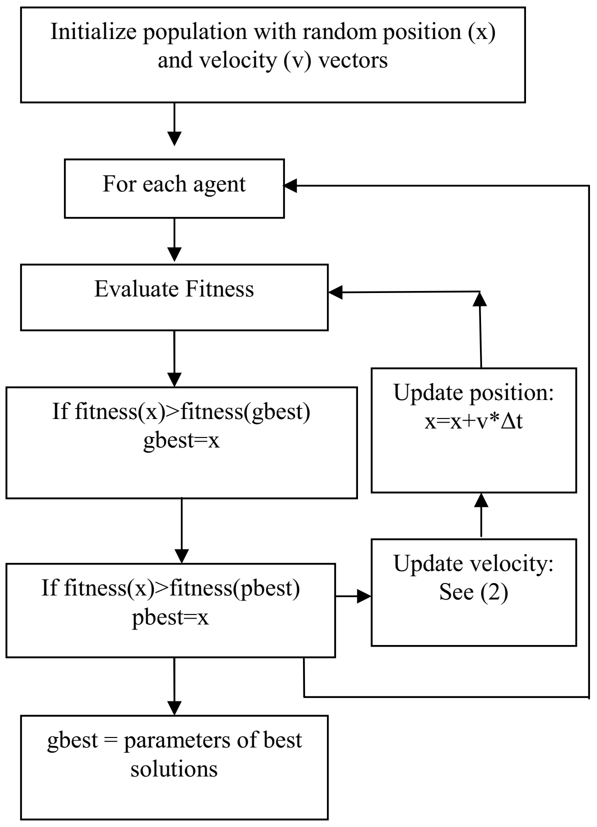
Initialize population with random position (x) and velocity (v) vectors
For each agent
Evaluate Fitness
If fitness(x)>fitness(gbest) gbest=x
Update position: x=x+v*Δt
If fitness(x)>fitness(pbest) pbest=x
Update velocity: See (2)
gbest = parameters of best solutions

Figure 5. PSO Algorithm.

B. Solve the optimized controller

The following Figure 6 shows how PSO technique effectively solves the Airship flight control problem [10] and [11]. By using the PSO algorithm, there is no need to compute the controller u; instead it was generated by the particles and updated using the method shown above. By setting a suitable search space range, which is one of the key processes to get the optimized result using PSO, particle numbers and also parameters c1, c2, and ω, the controller for the airship will fast converge to the desired solution.

We have already got the trained airship neural network model in section II; we now combine it together with PSO method. Every step's controller generated by PSO will come into the airship model with next step's inputs, and the output will be used to update the neural network model then by PSO algorithm we update the next step's controller. In this way, we got a dynamic airship model and a real time optimized controller as well. The following flow chart showed in Figure 6 is how we find the real time optimized controller for the dynamic airship model based on PSO algorithm.

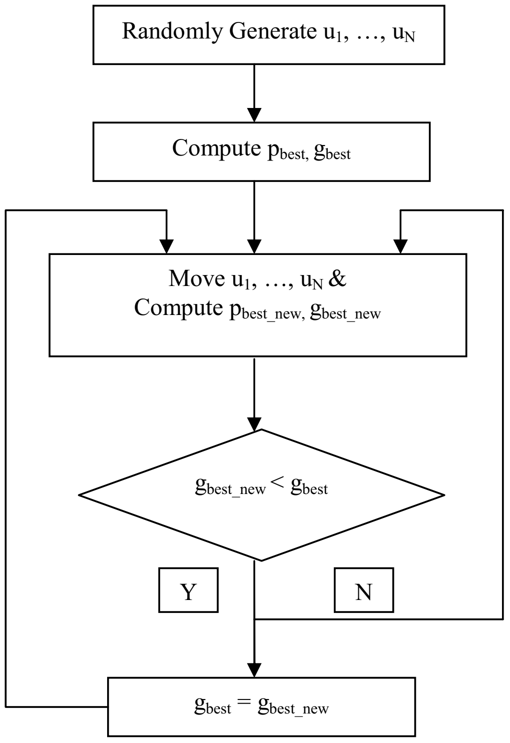
Randomly Generate $u_1, \ldots, u_N$
Compute p_{best}, g_{best}
Move $u_1, \ldots, u_N$ & Compute $p_{best\_new}, g_{best\_new}$
$g_{best\_new} < g_{best}$
Y
N
$g_{best} = g_{best\_new}$

Figure 6. PSO Based Optimized Controller.

Particle Swarm Optimization will converge quickly with adding some constraints [12] and [13]. From the above flow chart, we can see this program using a 10 steps predict controll. With the help of the predict control the converged speed incresed a lot. The reason being is PSO method does randomly searching, we need add something to eliminate the random factors.

IV. Simulation Results

A. PSO based controller for linear airship model

The requirements for the airship works as a temporary mobile station or a surveillance of the surface topography airship are either maintains a fixed position or a stable velocity. Therefore the controller we need is either to stabilize the airship at a certain height or a fixed velocity. At beginning of the program, give initial states values to the longitudinal model together with particle swarm optimization first time's randomly generated controller. After getting output from the model, we check whether the state (either position or velocity) we want to control already reached our reference value. If yes, then save the controller and stop. If it is not, then go to the next iteration and let PSO find a better controller. Since the fitness function we set up for the optimal control here is: $F = F + (x-d)^2$, x is our current position, obtained by particles and recorded every iteration, and d is the reference position, we give F a zero initial, our goal is to let x go to d, so which means in PSO method, we just minimize the result of F. By doing this we reformulate the airship control problem to a regular PSO optimized problem. Figure 7 shows how the airship holds on a fixed position based on the linear longitude model. Reference position is 10.

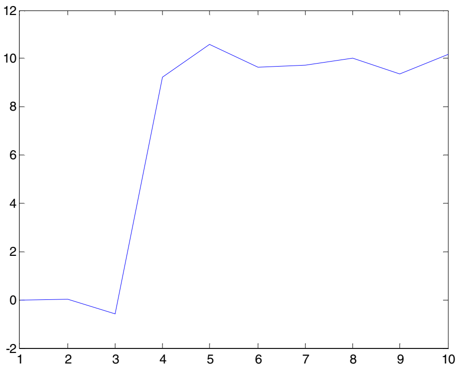

Figure 7. Airship position holds mode based on linear model.

It is the same thing for a velocity control mode. The only thing we need to change is for the fitness function, we change it to: $F = F + (u - d)^2$, u is the current velocity obtained by particles and recorded every iteration, and d is reference of velocity. The PSO's goal is still to minimize the F function so that u can go as close as to d. Figure 8 shows the airship holds on a fixed velocity based on the linear longitude model. The reference velocity is 5.

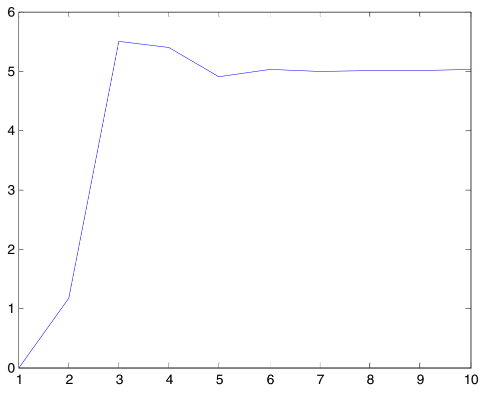

Figure 8. Airship velocity holds mode based on linear model.

From the above two figures, we can see the result for the velocity is better than the result for the position. The reason being is longitudinal velocity is longitude only. For the position it is affected with vertical and longitude dimension both. And because the model using here is linear longitudinal model, so it is not well exactly representing the vertical dimension. Therefore the result is better for velocity than position.

B. PSO based controller for neural network model

The above is the two hold mode based on linear longitudinal model. And the following are PSO optimal control based on real data neural network model. Figure 9 is the position hold result based on this neural network model. The real position data has a basement of 120. All the position data are above 120. Since all other 4 states are really small number, some of them are less than one. To get a better training result, we deduct a basement of 120 which make the position state goes to the same range as the other states. Then the neural network output becomes perfect match on all states. Figure 9 is the position hold with reference height 121.4. The basement is 120.

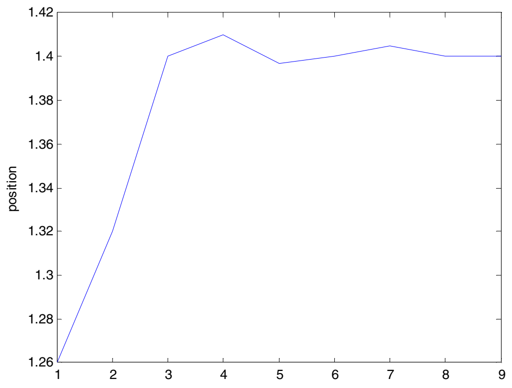

Figure 9. Airship position holds mode based on NN model.

The following Figure 10 is the online velocity hold based on decoupled neural network model. After the PSO give the optimal controller, we feed it back to update the previous neural network model. Then redo the PSO based optimal control. The following shows the online neural network model, the velocity hold mode result.

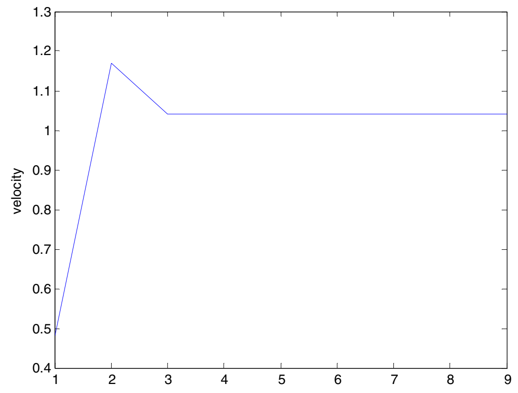

Figure 10. Airship velocity holds mode based on online NN model.

From the above result Figure 10, we can see even with the online model which is more complicated than the linear airship model, the PSO still give a really good result to either hold the airship for a fixed position or speed. The following Figure 11 is the airship online model states simulation with the PSO controller. Without applying the controller, the longitude velocity will have a significant change together with other states.

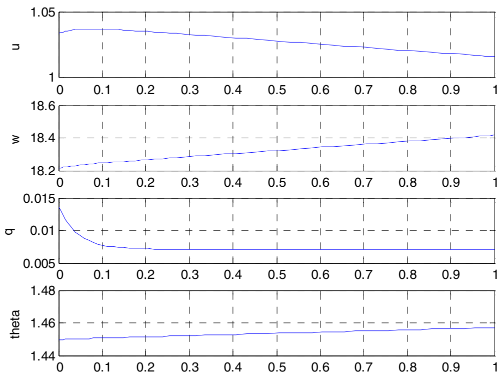

Figure 11. The Online Model with the PSO Controller

V. Conclusion

This paper develops the controller design on the flight control of Tri-Turbofan airships based on particle swarm optimization techniques. Two primary requirements for the surveillance airship is the ability to maintain either a position or a velocity hold mode. This paper uses the online neural network airship model to obtain real dynamic results which will help particle swarm optimization algorithm to find a more accurate real time optimal controller. This paper examines a real-time optimal control method making use of a particle swarm optimization technique coupled with a neural network online updating method and a prediction stage to develop a series of control laws to meet the constraints of the optimal problem while maintaining either the velocity or position set point.

Acknowledgment

This work was supported in part by the U.S. National Science Foundation under grant ECS-0554117 and EEC-0649172. The authors would like to gratefully acknowledge the use of the equipment and personnel at the University of Texas Health Science Center at San Antonio GAIT Laboratory during the airship flight testing.

References

[1] M. Frye, S. Gammon, and C. Qian, ”The 6-DOF Dynamic Model and Simulation of the Tri-Turbofan Remote-Controlled Airship,” to appear in the preceedings of, 2007 American Control Conference.

[2] J. Kennedy and R. C. Eberhart, “Particle swarm optimization,” in Proc. IEEE Conf. Neural Networks IV, Piscataway, NJ, 1995.

[3] S. M. Gammon, M. T. Frye, and C. Qian, “The Development of the Tri-Turbofan Airship Model for Autonomous Flight Control Research,”Proceeding of the 2006 AIAA Guidance, Navigation and Control Conference,Workshop, and Exhibit, Keystone, Colorado, August 2006.

[4] S.B.V. Gomes and J.G.Ramos, "Airship Dynamic Modeling for autonomous Operation," Proceeding of the 1998 IEEE International Conference on Robotics and Automation, Leuven, Beligium, May 1998, pp. 3463-3467.

[5] C.P.Burgess, Airship Design, The Ronald Press Company, Ney York; 1927.

[6] G. A. Khoury (ed.) and J. David Gillett (ed.), Airship Technology, Cambridge University Press, Cambridge, England; 1999.

[7] R. C. Nelson, Flight Stability and Automation Control, McGraw-Hill Book Company, New York; 1989.

[8] R. C. Eberhart and Y. Shi, “Evolving artificial neural networks,” in Proc. 1998 Int. Conf. Neural Networks and Brain, Beijing, P.R.C., 1998.

[9] J. Robinson, S. Sinton, and Y. Rahmat-Samii, “Particle swarm optimization in electromagnetics” in Proc. IEEE Transactions on antennas and propagation, vol. 52, NO. 2, February 2004

[10] J. Robinson, S. Sinton, and Y. Rahmat-Samii, “Particle swarm, genetic algorithm, and their hybrids: optimization of a profiled corrugated horn antenna,” in Proc. IEEE Int. Symp. Antennas Propagation, vol. 1, San Antonio, TX, 2002, pp. 314–317.

[11] R. C. Eberhart and Y. Shi, “Particle swarm optimization: developments, applications and resources,” in Proc. 2001 Congr. Evolutionary Computation, vol. 1, 2001.

[12] M. Clerc and J. Kennedy, “The particle swarm-explosion, stability, and convergence in a multidimensional complex space,” IEEE Trans. Evol. Comput., Feb. 2002.

[13] N. Lovbjerg, T. K. Rasmussen, and T. Krink, “Hybrid particle swarm optimizer with breeding and subpopulations,” in GECCO-2001, p. 469.